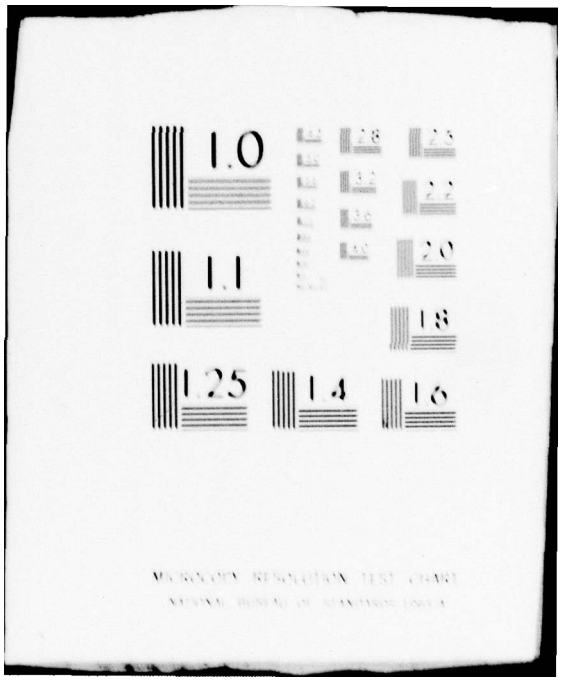
AD-A075 106 NAVAL RESEARCH LAB WASHINGTON DC F/G 9/2
THREADER MICROCOMPUTER ANTENNA CONTROLLER.(U)
SEP 79 L E RUSSO
UNCLASSIFIED NRL-MR-4067 NL

1 OF 2
AD A075106


1.0
1.1
1.25
1.4
1.6
2.8
2.5
2.2
2.0
1.8

NRL Memorandum Report 4067

Threader Microcomputer Antenna Controller

LEONARD E. RUSSO

Signal Exploitation Branch
Communications Sciences Division

September 24, 1979

NAVAL RESEARCH LABORATORY
Washington, D.C.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NRL Memorandum Report 4067	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): THREADER MICROCOMPUTER ANTENNA CONTROLLER	5. TYPE OF REPORT & PERIOD COVERED: Final report on the NRL problem.
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): Leonard E. Russo	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Research Laboratory, Washington, D.C.	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NRL Problem R16-43.201, WR-7-9225
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Electronic Systems Command, Washington, D.C. 20360	12. REPORT DATE: September 24, 1979
	13. NUMBER OF PAGES: 97
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microprocessor
Microcomputer
8080
Antenna controller
Digital control
Intelligent control
Intel SBC80 modules

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The THREADER microcomputer antenna controller is the electrical interface and controller between the THREADER PDP11 and SA antenna pedestal. The antenna controller is slaved to the PDP11; however, the controller contains its own microprocessor, memory and program and can respond to a suite of complex commands. The controller responds to ASCII commands from the PDP11, sent over an RS232 line. The antenna controller also provides the software/hardware interface to a Time Code Reader for system time. The controller is configured from off the shelf hardware, and is implemented as a software program. The programmability and generality of the antenna controller make it potentially useful in other control applications.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

APPENDICES

THREADER MICROCOMPUTER ANTENNA CONTROLLER

FIGURES

TABLES

APPLICABLE DOCUMENTS

1. Datum Inc., "Instruction Manual-Model 9200 Time Code Reader Part. No. 9200-715", Anaheim, Ca.

2. Intel Corp., "SBC80P20 User's Guide", Order No. 98338, Santa Clara, Ca., 1976.

3. Intel Corp., "SBC80/20 and SBC80/20-4 Single Board Computers Hardware Reference Manual", Order No. 98317c, Santa Clara, Ca., 1977.

4. Intel Corp., "SBC104/108 Combination Memory and I/O Expansion Boards Hardware Reference Manual", Order No. 98277b, Santa Clara, Ca., 1976.

5. Intel Corp., "SBC519 Programmable I/O Expansion Board Hardware Reference Manual", Order No. 98385a, Santa Clara, Ca., 1977.

6. Intel Corp., "SBC635 Power Supply Users Manual", Order No. 98298a, Santa Clara, Ca., 1976.

7. National Semiconductor Microcomputer Systems. "Series/80 BLC016 16k RAM Board", Pub. No. 420305375-001a, Santa Clara, Ca., 1977.

8. Scientific Atlanta, "Model 1848 Digital Comparator Instruction Manual, 3rd. Edition", Atlanta, Ga., Feb-71.

9. Karp, Tony, "Develop Software with Flowgrams", Elec. Design, Vol. 26, No. 16, 2 Aug. 1978, p. 84.

10. Intel Corp., "8008 and 8080 PL/M Programming Manual", Rev. A, 1975.

11. DECUS Program Library, "BLISS11 Programmer's Manual", Decus No. 10-213,11-160, April 1974.

THREADER MICROCOMPUTER ANTENNA CONTROLLER

1.0 Introduction

The THREADER antenna controller is a programmable instrument designed to control the Scientific Atlanta(SA) series 3000 pedestal and drive electronics. Originally conceived as a replacement for the paper tape input available with the series 3000 system, the antenna controller's capabilities go far beyond those of the paper tape system. The controller performs functions in response to a set of commands which are upward compatible with those of the old paper tape system.

The salient features of the antenna controller are:

i) Control by an 8080 microprocessor.
ii) Local memory available.
iii) Upward compatibility with the original paper tape system.
iv) Communication with a computer/terminal via a serial RS232 line.
v) An extensible command set composed of ASCII characters.

The controller will respond properly to a paper tape read through a reader with an RS232 interface, but the command set is a superset of the paper tape system commands allowing remote operation and system monitoring. As the controller contains memory, it is possible to log information about the antenna pedestal or to transfer the information directly over the serial interface to the console. In the case of the THREADER system the console is the Keyboard/CRT display of the PDP11 system. The antenna controller has added control to facilitate communication between the PDP11 and 8080 computers.

Manuscript submitted June 29, 1979.

2.0 System Hardware Configuration

The THREADER antenna control system(Fig. 1) consists of the following equipment:

i) GT44 system including a PDP11/40.
ii) SBC based microcomputer.
iii) Time code reader(TCR).
iv) Antenna pedestal and drive electronics.

The PDP11 provides the graphics display and console for the operator, logs system messages, generates pointing data from ephemeris bulletins, and provides a data base for tracked vehicles. The PDP11 also controls the microprocessor antenna controller via commands across an RS232 serial channel.

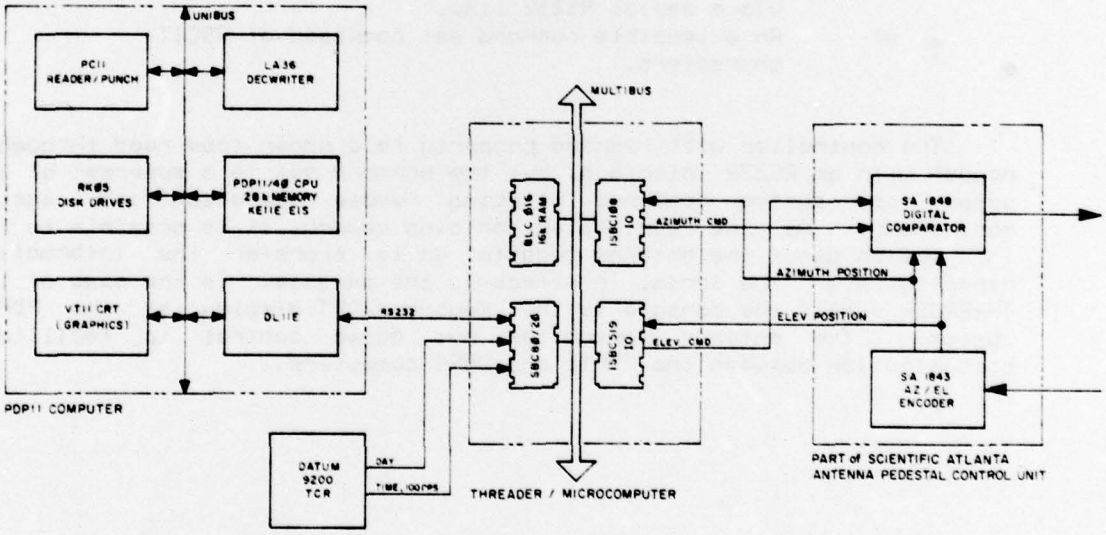


FIGURE 1. THREADER Antenna Controller(with PDP11 shown)

The microprocessor antenna controller consists of an 8080 microprocessor, memory and interfaces to the PDP11, antenna pedestal drive electronics and time code reader. The controller is implemented in software on a microcomputer composed of SBC-type modules. SBC modules are standard boards supplied by the Intel Corp. and others to perform computer system functions , eg. memory, I/O, CPU, etc. All SBC modules are 6.75x12 inch printed circuit boards which communicate with a standard bus, the Multibus, as defined by the Intel Corporation. These modules are effectively an industry standard for microcomputer systems.

A wide variety of modules exists, many of which are second sourced. The configuration of SBC modules used for the antenna controller is given in Table 1.

TABLE 1. SBC Modules in Antenna Controller

	Component	Description
1.	System 80/20	SBC 80/20 Microcomputer, backplane, chassis, power supply.
2.	SBC 80/20	Part of System 80/20. 8080 microprocessor, 2 parallel I/O ports, RS232 port, 8-level priority interrupt, system monitor, realtime clocks.
3.	SBC 108	8k bytes RAM, 2 parallel I/O ports, RS232 port.
4.	BLC 016	16k bytes RAM.
5.	SBC 519	3 parallel I/O ports.

3.0 System Interfaces

The antenna controller interfaces to the DATUM 92 00 time code reader and to components of the SA3000 pedestal electronics. The PDP11 is linked to the antenna controller via an RS232 serial interface port on the SBC 80/20 module.

The function of the antenna controller is to respond to commands from the PDP11, transmitting command information on to the Scientific Atlanta system or supplying information from the SA system or the DATUM time code reader. As the antenna controller contains memory, command or request information m ay be stored locally as needed.

3.1 SA3000 Pedestal Interfaces

The antenna controller interfaces with two components of the SA3000 system: the SA1848 Digital Comparator and the SA1843 Synchro to Digital Convertor. Azimuth and elevation commands output from the antenna controller go to the SA1848 Digital Comparator. The SA1848 compares the digital position command from the antenna controller with the current position of the pedestal output from the SA1843 Synchro to Digital convertor. The result of the comparison produces an analog signal which drives the 3000 pedestal to the position commanded by the controller. The antenna controller has a command output and positiDn input port for both azimuth and elevation. The azimuth/elevation cables from the SA1843 are tapped and connected to controller input ports to provide position information. Both azimuth interfaces reside on the SBC 519 module: both elevation ports reside on the SBC 108 module.

3.2 Time Code Reader Interface

Two parallel interfaces were used to interface the Datum 9200 Time Code Reader to the microprocessor. Both interfaces are on the SBC80/20 microcomputer board. The TCR can read BULLSEYE or IRIG-B code: however, the system reads IRIG-B as data is supplied in this code.

One parallel interface, located on the SBC80/20 microcomputer board, provides hours through seconds as well as a 100pps pulse train. The pulse train is used by the SBC80/20 h ardware to generate the .1 sec. tick used as the system event clock.

Another input interface on the SBC80/20 is used to read the Julian date from the TCR. The antenna controller may thus read absolute time, days through seconds, from the TCR.

The PDP11 may request time from the antenna controller. The request is honored on the next second boundary; the time sent is accurate to within the time needed to transmit the ASCII time string over the channel(about 60 msec. at 1200 bd.).

3.3 Serial Interface

Communication with the antenna controller is via a serial RS232 channel which may be operated at any baud rate from 110 to 9600, and is nominally run at 1200 baud. The serial line goes from a DL11E on the PDP11 Unibus to an 8251 USART on the SBC 80/20 microcomputer board. Once a handshake has been established between the PDP11 and the SBC80/20 port, control lines are not altered. Channel control is accomplished with the use of ASCII control characters. All communication over the channel is done using ASCII characters.

As all data over the channel is ASCII encoded, and the channel is a standard RS232 channel, any device(minicomputer, terminal, paper tape reader) with an RS232 interface and ASCII character set should be able to talk to the antenna controller.

4.0 Data Structures

The antenna controller operates on data transmitted over the serial channel. The data consists of a subset of the ASCII character set configured as commands, requests, records and files. Understanding the data structures is a prerequisite to understanding controller operating modes, command interpretation and software.

The character set used by the antenna controller is the following: A,E,I,J,K,L,M,O,R,T,V,W,Z,0,1,2,3,4,5,6,7,8,9. In addition, certain punctuation[.,:,+,-,',',] and control characters are used.

4.1 Command Structure

Commands are formed from ASCII encoded letters and numerals. Commands all have the same format: an ASCII letter followed by an optional ASCII numeric string of the form: sddd.d, where 's' is '+', '-' or null, and 'd' is a digit. The form of the numeric string is quite flexible; for example, A54.0,A+54.,A54,A+54.0 would all be translated by the controller to the operation 'point the antenna to 54 deg. in azimuth'.

There is one exception to the normal command format. The T-command requires a 6-digit string supplying time in hours through seconds(hhmmss).

4.1.1 Requests

A request for information is a special type of command. The format for a request is an ASCII 'L' followed by the command letter, eg. the request LA(=list azimuth) would return A054.0 if the antenna were pointing 54. degrees in azimuth.

4.1.2 Command Suite

For each command, there is a corresponding request, although sometimes the command or request may be a null operation. The command suite with brief description is given in Table 2.

TABLE 2. Antenna Controller Commands

	Command	Request	Description
1.	A	LA	set/list azimuth
2.	E	LE	set/list elevation
3.	I	LI	set/list interpolation interval
4.		LJ	list Julian date
5.	K		abort autorun, return to immediate mode.
6.	M		enter immediate mode
7.	O	LO	set/list time offset
8.		LR	send report file
9.	T		set autorun start time
		LT	list realtime to console
10.	V	LV	set/list elevation offset
11.	W		enter/exit waitmode
12.	Z	LZ	set/list azimuth offset

4.2 Records

One or more commands are concatenated to a suffix ':' to form a record. The record is the processing unit recognized by the controller, ie. processing a complete record is a software task. The record format is upward compatible with the paper tape format. A ',' is an optional command separator within a record. A record containing no commands serves as a timing marker.

4.3 Files

Two types of files are transmitted over the serial channel: data files and report files. These files are framed by special control characters to alert the receiving computer that a file is being transmitted.

A data file is a group of records transmitted from the PDP11 to the antenna controller. The data file contains pointing commands generated from the ephemeris bulletins; the file may contain other commands. The data file is stored in th e antenna controller data buffer to be used as the source of command records for an autorun.

A report file is a group of records containing information about an autorun track. The information requested by requests in the data buffer is stored in the antenna controller report buffer on a record by record basis. For each data buffer record, there is a corresponding report buffer record. The report buffer is transmitted by the antenna controller over the serial channel in response to the R-request after an autorun is completed. The contents of the report buffer framed by the appropriate control character and sent over the channel constitutes the report file. Outputting a report file is the lowest priority software task; however, once a request for report is honored, character output is performed until tht entire report buffer is emptied, aprocess taKing 10-20 sec. at 1200 baud[no request for report is issued by the PDP11 in the current THREADER system].

5.0 Operating Modes

There are two basic modes of operation in the antenna controller: normal or immediate mode and autorun or remote mode.

5.1 Immediate Mode

Immediate mode is the default mode. When the controller is in immediate mode, it queues commands from the PDP11 to a circular input buffer. Processing commands from the circular input buffer is a system task. As soon as this task has priority, the command is processed and the requested information, if any, is queued to the circular output buffer. Outputting data from the circular output buffer is also a task. The output information will be transmitted over the serial channel to the PDP11 once the output task is activated.

Immediate mode is always active; however when the antenna controller is in remote mode, some commands which would disturb the remote operation are forbidden and are treated as null commands. Immediate mode provides a means to monitor the antenna controller state at all times from the PDP11 console.

5.2 Autorun Mode

The autorun mode may be entered by issuing the immediate mode T-command. Autorun begins after the start time supplied in the command. Once initiated, the controller begins reading commands from the data file stored in the local memory data buffer. The data file contains pointing and other commands downloaded from the PDP11 to the microcomputer prior to the commencement of remote mode. Thus, the controller, once primed, can track and log information on a track without further intervention from the PDP11, unless intervention is required.

When remote mode is initiated, a countdown begins to the start time. The antenna is prepositioned by performing the commands in the first record in the data buffer. Once the run begins, command records are read at intervals specified by the I-command which gives the interval between pointing records to the nearest .1 second. A command from the PDP11 sets the interval. Typical record spacing is 10,20 or 40 seconds. As the available memory in the microcomputer limits the fineness of pointing record spacing, the controller interpolates azimuth and elevation between pointing records.

The progress of an autorun track may be examined by immediate mode

requests from the console or by requests stored in the data buffer. The responses to requests stored in the data buffer are stored in the report buffer. For each data buffer record, demarcated by ':', there is a corresponding report buffer record, demarcated in like manner. At the end of autorun, the report file may be sent to the PDP11. The report option is not utilized by the PDP 11 in the current THREADER system.

When autorun begins, a '^A' is sent to inform the PDP11 that the microcomputer is in autorun mode; when autorun ends, a '^Z' is sent to inform the PDP11 that the microcomputer has returned to immediate mode operation.

6.0 Antenna Controller Operation

6.1 Intercomputer Communication

Several ASCII control characters are used in intercomputer communication between the PDP11 and the SBC80/20 processor. These are generally intercepted immediately by the interrupt software in both computers and cause subsequent data over the channel to be interpreted in a different manner. Control characters are indicated by a prefix '↑', eg. '↑A' is a control-A.

All characters read by the antenna controller from the channel are put into a 256-byte circular input buffer. If the controller were heavily loaded with processing tasks, it is possible the buffer might overflow. A '↑Q' is sent over the channel to notify the PDP11 of imminent buffer overflow. When the microcomputer empties the buffer by processing command records so that overflow is no longer a danger, the PDP11 is sent a '↑S'.

Data files, ie. those files containing many records to be stored in the antenna controller memory are framed by '↑B' and '↑D' so that a data file looks like '↑B.....(records)...↑D'. Data files provide the commands to control the antenna controller during a remote run, ie. when it is moving the antenna along a track independently of direct commands from the PDP11.

Report files are logs of information about a track stored as a result of commands in the data file. The antenna controller would ,on request, send the report file to the PDP11 in the same format as the data file, ie. '↑B...(records)...↑D'[this feature not implemented in THREADER system].

When the antenna controller starts a remote run, it informs the PDP11 of the event by sending '↑A'. Similarly, when the remote run is completed, the PDP11 is informed by '↑Z'.

'↑X' is a debug feature which returns control of the microcomputer to the SBC monitor. A summary of control character usage is given in Table 3.

TABLE 3. ASCII Control Characters

Character	Function
^A	Notify PDP11 of start of autorun.
^B	Signifies beginning of datafile.
^D	Signifies end of datafile.
^Q	Notify PDP11 of imminent input buffer overflow.
^S	Notify PDP11 danger of buffer overflow is past.
^X	Exit and return to SBC monitor.(Debug only!)
^Z	Notify PDP11 autorun complete.

6.2 Description of Commands

The current command set of the antenna controller will be described in this section. Not all commands are utilized in the current THREADER system. The command structure is such that it is easily extensible. Commands may be grouped as pedestal control(A,E), autorun control(I,K,T), offsets(O,V,Z), altering controller state(M,W) and requests for special information(J,R,T).

The A-command and E-command point the antenna in azimuth and elevation, respectively; the corresponding requests list the current antenna position.

For an autorun to work properly the time interval between pointing records must be specified exactly. The I-command provides a means to set and list this interval. If it is necessary to end autorun, the K-command will return the controller to the immediate mode; no numeric string is required with this command. The T-command loads the start time into the microcomputer. As time is specified as a six digit string: HHMMSS, the normal format for the numeric string is not used; instead six digits provide time-of-day. Setting time begins a countdown to the start time; the first record of commands in the data buffer is read to preposition the antenna. After start time, records are read at each interval mark and position is interpolated at .1 second intervals. Interpolation to .1 sec. causes the antenna to track in a quasi-continuous manner; abrupt starts and stops and ensuing vibrations are minimized. Note that the 'T' and 'LT' commands are not complementary in the manner of, for example, 'A' and 'LA'.

Offsets may be set in time, azimuth and elevation. Azimuth offset is set by the Z-command; elevation o ffset is set by the V-command. Azimuth and elevation offsets are added to position information after interpolation. Offsets are zeroed at the start of an autorun.

Time offsets, set by the O-command, either delay(positive numeric string) or advance(negative numeric) the reading of records with respect to real time. If the antenna were tracking a satellite accurately, a delay would allow the satellite to advance beyond the antenna's pointing position by delaying antenna movement for a prescribed period; an advance would advance the antenna's pointing position further along the track than the satellite. The request for time offset yields zero value unless a delay is active, in which case the remaining delay time in seconds is returned.

The M-command returns the system to manual mode while letting the autorun calculations progress without any output. The M-command will support future system features and should not be used.

The W-command alters the mode of pointing in immediate mode. A W-command followed by a non-zero numeric activates the wait mode. In the wait mode, all positioning commands are held until the antenna comes within .5 deg. of the position. This mode thus prevents the antenna from slewing rapidly in response to a succession of commands; instead the antenna is allowed to settle.. before the next command is executed. The W-command allows the operator to step the antenna discretely. The command 'W0.0' exits waitmode.

Several commands are available only to supply information requested by the PDP11. The J-request and T-request return the Julian date and realtime, respectively. The R-request empties the report buffer of the last autorun completed.

In the THREADER system, the user never issues commands directly to the microprocessor system. The user commands the PDP11 from the graphics display console and the PDP11 subsequently generates and sends the appropriate command to the microprocessor. The PDP11 uses the following subset of the total command suite of the antenna controller: [A,E,I,J,K,T,V,Z].

7.0 Program Software

The antenna controller is based on a 8080 microcomputer system. As such, the microprocessor software defines the controller operation. The software is written in a high-level language which compiles on a PDP10. The high-level languge is PL/M[Ref. 10], a structured language which generates 8080 code. During program development, the hex-object file is downloaded from the PDP10 to the PDP11 and stored on PDP11 disk. Prior to initiation of THREADER program, the object file must be downloaded from the PDP11 to the microcomputer memory using SBC monitor commands. Future versions of the antenna controller may have the controller program in Read-only memory(ROM).

The program software consists of sev en blocks:

i) Utility routines
ii) Processor setup routines
iii) Data movement routines
iv) Interpolation routines
v) Command Decode and Processing routines
vi) Interrupt Service Routines
vii) Task Priority Resolver

The utility routines handle character encode, character decode and BCD arithmetic operations for two and three byte numbers.

Processor setup routines initialize flags, semaphores and counters. Parallel I/O must be set up for input or output, number of bits, mode, etc. The interrupt jump table for the priority interrupt structure must be loaded. Some initialization is done by the SBC monitor prior to entering the antenna controller program.

Data movement routines generally move information to buffers, the serial channel or I/O ports. Reports, antenna positioning and record output triggering are procedures within this block.

Interpolation routines support the interpolation between autorun pointing records. Computing the interpolation deltas as each new record is read, doing the actual interpolation, and handling anomalous conditions (crossing 360-0 deg. boundary, interpolation beyond 90 deg. elevation, etc.) are processes performed by procedures in the interpolation block.

Command decode and processing is the largest block of code in the program. A heirarchy of procedures is called which locates and determines the command within the record, decodes the numeric data associated with the command and allows the command to be executed on the condition that the mode of operation of the antenna controller(autorun, setup-autorun, immediate) will not be disturbed. Commands will be interpreted in different manner or not at all depending upon the controller mode.

7.1 Interrupts

Interrupt service routines exist for system time(0.1 second tick), character input and character output.

7.1.1 System Timing

The system timing interrupt service routine, TICK, has highest priority and controls countdown to autorun, reading of autorun pointing records and transfer of real time to the PDP11. Note that time transferred is accurate to within the time needed to send the seven character time string over the serial channel(about 60 msec. at 1200 b d.).

7.1.2 Character Input

The character input service routine, PDPSBCISR, inputs characters into the appropriate buffers and handles control characters : ^B,^D,^Q, used for data file transfer and channel control. Another control character, ^X, is used to exit to the SBC monitor for debug purposes.

If the character is a control signal, the appropriate flags are set immediately and the character is discarded; otherwise the character is stored in the appropriate buff er and the buffer pointer is incremented. If the character is the record demarcator, ':', the record count is incremented. If a character is part of a data file, it is stored in the data buffer in the controller memory. Otherwise, the character is stored in the circular input buffer.

7.1.3 Character Output

Character output is by software polling, but interrupt driven character output may be implemented in a future version of the controller program.

Character output is dependent on the software task priority.

Higher priority tasks can override character output. Output is by record. A special routine primes the output of the record by outputting the first character to the RS232 output port buffer. Subsequently, the output buffer is polled for buffer ready, provided there are no higher priority tasks. A new character is loaded for output eve ry time the output buffer is empty and the character output routine has highest priority. The output process terminates when the entire record has been output.

7.1.4 Priority of Interrupts

The priority of interrupts is(highest to lowest): SBC monitor interrupts, system 'tick', and character input. The character output interrupt service routine, though present, is not used in this version of the antenna controller software. Character output is a polled task in the c urrent software.

7.2 Task Priority Resolver

The microprocessor antenna controller is task oriented. Every time the microprocessor is idle, a set of conditions is checked to see if there are any pending tasks. The tasks are ordered on a priority basis. If two or more tasks are pending, the highest priority task is executed. Once the task is completed the microprocessor is idle and the set of conditions is checked again, reinitiating the process just described. Tasks associated with autorun have highest priority. Next highest is informing the PDP11 that there is no longer danger of input buffer overflow in the microcomputer. Character output has priority over processing immediate mode records. an ordering which should make overflow of the circular output buffer unlikely. Sending the report file to the PDP11 is the lowest priority task in the system.

The implementation of the task resolver makes it easy to add new tasks of lower or higher priority to the software.

7.3 Program Statistics

The software program is just under 7400 bytes of machine code and requires just under 5100 bytes of memory for data storage, including the report and data buffers of 2048 bytes each. The program origin is hex-location (4003). The PL/M source code listing is included in Appendix E.

THREADER MICROCOMPUTER ANTENNA CONTROLLER

APPENDICES

THREADER MICROCOMPUTER ANTENNA CONTROLLER

APPENDIX A

Hardware Modifications

A.0 Introduction

Several hardware modifications were made in the implementation of the antenna controller. Hardware modifications are needed to choose interrupt structure, parallel port input-output, to generate clock 'tick' and enable certain RS232 control lines.

A.1 SBC 80/20 Modifications

Most modifications were made on the SBC 80/20 board. Wherever possible, modifications were localized on the board which is the microcomputer in the antenna controller. Schematic diagrams for the SBC80/20 may be found in Ref. 3, Appendix A.

A.1.1 Generation of the 0.1 Second Pulse 'Tick'

The generation of the 0.1 second event clock was done with hardware and software. A 100 pps signal from the time code reader was input to the SBC 80/20 port E6. The signal was then jumpered to counter1(CTR1) of the 8253 triple interval counter. A software routine sets the mode of CTR1 to interrupt on terminal count, thus setting a flag which is input to the priority resolution circuitry. This flag remains set until the counter is reloaded in the 'tick' interrupt service routine. Loading '9' into the counter causes the interrupt flag to be set every 0.1 second for the 100pps input signal. The modifications from the default wiring to cause the 100pps signal to be input to CTR are shown in Table A1 (See SBC 80/20 Hardware Manual, Appendix A).

TABLE A1. Modification to Generate 0.1 Second Tick
[Reference 3, A11-A12]

Delete	Add	Comment
142-141		Remove phi-2 input from CTR1
54-55		Remove 100 pps from port E6
	54-141	Add 100 pps to CTR1 input

A.1.2 Interrupts

A variety of potential interrupt are strappable into the priority interrupt circuitry. System interrupts for restart and SBC monitor operation were left as the highest priority interrupts. The order of priority following the monitor interrupts was: 'tick', character input and character output [not implemented]. Table A2 shows the interrupt structure for the current antenna controller. The referenced signals may be found in the SBC 80/20 hardware reference manual. Note that interrupts five through seven are disabled in software. IR0 is highest priority; IR7is lowest.

TABLE A2. Interrupt Assignment
[Reference 3, A13]

Interrupt	Pin	Signal	Pin	Description
IR0	24			
IR1	25	INT2	45	Monitor reset
IR2	26	OIT0	35	Monitor Single Step
IR3	27	OIT1	34	0.1 sec. Tick
IR4	28	RXR	41	Character in ready
IR5	29	TXE	32	Character out ready
IR6	30			Spare
IR7	39-36	GND		Spare

A.1.3 RS232 Serial Interface

The serial RS232 channel runs from the 8251 USART(Universal Synchronous/Asynchronous Reciver Transmitter) to the DL11E serial port interfaced to the PDP11 Unibus. The PDP11 thinks it is a data terminal and the SBC 80/20 is wired as a data set. As the DL11E recognizes the 'Data Carrier Detect'(DCD) signal but not the 'Data Set Ready'(DSR) signal, the DSR output from the USART was jumpered to the DCD line so that the PDP11 could register a response to the setting of DSR in the antenna controller port. To effect this connection, a line was connected from A16-8 to A17-14 on the SBC 80/20 board[Ref. 3, A12].

A.2 Parallel Port I/O Setup

Parallel I/O is determined by a combination of hardware and software. The appropriate software is given in the procedure 'SETIO' in the controller code. A parallel group is a set of three 8-bit ports on a common edge connector having common control. Each group requires four IO addresses: one for control, one for each port. Looking at the populated side of an SBC board, group 1 is on the left. Ports are also locatable by the last hex digit in the port address embossed on the particular SBC board. Ports not set up in software default to 24 bit input ports.

Interface to the time code reader is handled on the SBC 80/20 board. The SBC 80/20 contains two parallel I/O groups[Ref. 3, A11]. Group 1 recieves time(hhmmss) from the time code reader; group 2 receives Julian date.

The SBC 108 board contains two parallel I/O groups[Ref. 4, A11-A12]. All azimuth information is handled via ports on this board. Groupl is the azimuth input port from the SA1843 Synchro to Digital convertor. Group 2 is the azimuth command output to the SA1848.

The SBC 519 board contains three parallel I/O groups [Ref. 5, A3]. All elevation information is handled via ports on this board. Group 1 is the elevation input from the SA1843. Group 2 is the elevation output

to the SA1848. Group 3 comprises three spare parallel ports.

Table A3 provides a summary of parallel port I/O. Input ports are of two types: those terminated by 8226 inverting transceivers and those terminated in SBC901 non-inverting pullup/pulldown resistor networks. Software corrects for the inversion at input ports terminated by 8226 transceiver chips. Output ports use 8226 transceiver chips and 7400 NAND gates as drivers.

TABLE A3. Parallel Port Input-Output

Location	IO Adr.	Mode	Driver	Function
SBC80/20 Gp. 1	E4	I	8226	seconds in
	E5	I	SBC901	minutes in
	E6	I	SBC901	hours in
	E7			control
SBC80/20 Gp. 2	E8	I	8226	date-low
	E9	I	SBC901	date-high
	EA	I	SBC901	
	EB			control
SBC108 Gp. 1	C4	I	8226	Azim. in-low
	C5	I	SBC901	Azim. in-high
	C6	I	SBC901	
	C7			control
SBC108 Gp. 2	C8	O	8226	Azim. out-low
	C9	O	7400	Azim. out-high
	CA	O	7400	
	CB			control
SBC519 Gp. 1	B0	I	8226	Elev. in-low
	B1	I	SBC901	Elev. in-high
	B2	I	SBC901	
	B3			control
SBC519 Gp. 2	B4	O	8226	Elev. out-low
	B5	O	7400	Elev. out-high
	B6	O		
	B7			control

A.3 Microcomputer Memory Configuration

The total memory in the THREADER microcomputer consisted of 8k ROM and 24k RAM. The memory map of the system is described in table A4.

TABLE A4. Memory Map for Antenna Controller

Location	Address Range	Mem. Type	Comment
SBC80/20	0-7ffH	ROM	SBC80/20 Monitor program.
	800H-fffH.	ROM	Unused.
	3800H-3fffH	RAM	Monitor tables. Programs to load interrupt jump table, interpolate to .01 deg.(at 3800H).
SBC108	(disabled)	RAM	Unused.
	(disabled)	ROM	Unused.
BLC016	4000H-7fffH	RAM	THREADER pgm.-4003H, program data-5df5H, memory boundary-7100h.

THREADER MICROCOMPUTER ANTENNA CONTROLLER

APPENDIX B

Backpanel Cabling for Azimuth/Elevation Ports

B.0 Backpanel Cabling for Az/El Ports

It was desired to keep the external cabling wirelist simple so that external cables could be fabricated using mass terminated, point to point, technology. To effect this simple interconnect, the cabling between the controller backpanel and the PC card edges on the SBC modules was wired according to the wirelist in Table B1 for all azimuth and elevation ports.

In connecting the backpanel to the SBC module, the even pins on the 3415 PC edge connector are always on top. The location of position and time ports is discussed in Appendix A. A glossary of signal names for the azimuth/elevation ports may be found in the SA1848 Digital Comparator manual[Ref. 8, pp. 2-4 to 2-6].

TABLE B1. Wirelist for Azimuth/Elevation Backpanel to SBC Cards

Signal	PC Connector Pin	Backpanel Connector Pin
0.01	23	1
0.02	21	2
0.04	19	26
0.08	17	27
0.1	47	3
0.2	45	4
0.4	43	28
0.8	41	29
1.0	39	5
2.0	37	6
4.0	35	30
8.0	33	31
10.0	15	7
20.0	13	8
40.0	11	32
80.0	9	33
100.0	7	9
200.0	5	10
400.0	3	34
SIGN	1	17
GROUND	50	50
RATE	25	19
CW	27	20
CCW	29	21
UPDATE	31	23

APPENDIX C

Time Code Reader Modifications

C.0 Time Code Reader Modifications

The DATUM 9200 time code reader(TCR) will read BULLSEYE and IRIG-B time codes. The unit has been modified to output Julian date. This addition was implemented by adding the optional circuitry for date output found in the 9200 manual. Circuit diagrams for the data option are enclosed in dashed lines in the TCR schematics. Date, as well as hours through seconds, is available for IRIG-B time code; the BULLSEYE provides only hours through seconds.

A new 50-pin connector designated J4 was added to the time code reader. Hours through seconds and 100 pps. signal were brought out through this connector. The connector type is 3M-3489. The 100 pps. signal was wired from location L2-8 to connnector J4-41[Ref. 1, VII-16].

All modifications for date output may be found in the 9200 manual. A wirelist for signal output on J4 and J5(the standard connector) are given in Table C1. Signal names may be found in Ref.1, IV-1.

TABLE C1. Time Code Reader Connector Pinouts

Signal	J4 Pinout	J5 Pinout
Sec1	49	1
Sec2	48	2
Sec4	47	3
Sec8	46	4
Sec10	45	5
Sec20	44	6
Sec40	43	7
Sec80	42	←
Min1	33	8
Min2	32	9
Min4	31	10
Min8	30	11
Min10	29	12
Min20	28	13
Min40	27	14
Min80	26	←
Hr1	37	15
Hr2	36	16
Hr4	35	17
Hr8	34	18
Hr10	38	19
Hr20	39	20
Hr40	40	←
100 pps	41	←
Gnd	24,25	35,36

Appendix C Time Code Reader Modifications

The cable from connector J4 to the antenna controller was mass terminated with 3M 3488 and 3415 connectors; therefore the wiring was point-to-point.

The cable from J5 to the antenna controller is not point-to-point. Connector J5 mates with cable connector P5, a 36-pin Amphenol 5730360 connector. The wirelist for cable connector P5 is given in Table C2. The cable connector on the antenna controller side is a Scotchflex 3M-3488.

TABLE C2. Date Cable Wirelist

Signal	P5	3M-3488	Comment
D1	21	31	Days
D2	22	32	
D4	23	33	
D8	24	34	
Ta1	25	43	Ten days
Ta2	26	44	
Ta4	27	45	
Ta8	28	46	
Ha1	29	47	Hundred days
Ha2	30	48	
Ha4	31	49	
Ha8	32	50	
Gnd	35	25	Ground

The cabling internal to the antenna controller, from the backpanel to the SBC 88/20 PC card edges was mass terminated, point to point.

THREADER MICROCOMPUTER ANTENNA CONTROLLER

APPENDIX D

Antenna Controller Flowgram

D.0 Antenna Controller Flowgram-Introduction

The flowgram[Ref. 9] provides an overview of program operation that is easily translated into the actual PL/M code. This ease of translation, along with the clarity of structured statements, makes the flowgram an attractive alternative to flow charts.

D.0.1 Flowgram Language

The syntax of the flowgram is typical of that associated with structured languages in general. The flowgram describes operations on or with program entities. Program entities include variables, arrays, flags, semaphores, and procedures. An entity is either an English word(eg. report) or a concatenated word with concatenation symbol '$'(eg. perform$command). Concatenation serves to clarify the purpose of an entity. All entities have analogs in the actual PL/M program. Another type of entity is a short sentences describing an operation which represents several lines of code(eg. request date).

The syntax of the flowgram is simple. A statement ends in a semicolon. A block is a group of statements enclosed in parentheses; the right parentheses serves a block delimiter as well as the end of the last statement within the block.

A standard 'If ... then .... else ....' statement is used. For example:

```
If immediate$mode then position azimuth
Else
    (Compute azimuth$delta; Form next
    azimuth output);
```

translates as:

```
If the state is immediate mode then position the
antenna in azimuth.
Else do these operations:
    Compute the azimuth delta.
    Form the next azimuth output.
```

Two 'Do' statements are used. In each case, 'Aj' may be a statement or a block. The first 'Do" statement is:

```
Do while <expression>;
    A1;
    A2;
    .....
    An;
End;
```

While the expression evaluates 'true', the operations A1 through An are performed in a loop. If the expression evaluates 'false', the loop is exited. A variant of the 'Do While <expression>' loop is the 'Do Forever' loop, in which the statements are performed cyclically forever because the implied expression is always 'true'.

The second 'Do' statement is the 'Do Case <expression>' statement:

```
Do Case <expression>;
    0:  A0;
    1:  A1;
    ......
    n:  An;
End;
```

For this statement, the expression will evaluate to a number between 0 and n. If the number is 'j' then 'Aj' is executed, and the 'Do Case' statement is exited. Finally, the symbol '<>' used in expressions means 'not equal'.

D.1 Antenna Controller Flowgram

```
Startup:
    Set$Interrupts;
    Set$IO;
    Zero$flags;
    /* task priority resolver */
    Do forever;
        If report$requested and no autorun and no
        current$output then priority=4;
        If input$records$ready then
            (If output$buffer not near full then
            priority=5);
        If current$output and transmitter$ready then
        priority=7;
        Else
            (If no pdp$bsy and output$record$ready then
            priority=6);
        If comm$line$off and no buffer$overflow then
        priority=8;
        If interpolation$ready then priority=9;
        If nxt$auto$record$ready then priority=11;
        If autorun$not$setup then priority=12;
        If autorun$active then priority=13;
        Do case priority;
            0:   ;
            1:   ;
            2:   ;
            3:   ;
            4:   Send$report;
            5:   Process$immediate$mode$record;
            6:   Initiate record$output;
            7:   Output$next$character;
            8:   Interpolate next az/el;
            9:   If all autorun$records processed then
                     (initialize data$buffer; output ^Z);
            10:  Process$autorun record;
            11:  Setup$autorun;
        End;
    End;
```

THREADER MICROCOMPUTER ANTENNA CONTROLLER
Appendix D Antenna Controller Flowgram

```
                        /* Setup Procedures */


Set$interrupts:
    Turn interrupts 5,6,7 off;
    Load interrupt$jump$table;
End Set$interrupts;



Set$IO:
    Set SBC188 Group1, SBC519 Group1 for 3-byte input;
    Set SBC188 Group2, SBC519 Group2 for 3-byte output;
End Set$IO;



Zero$flags:
    Initialize input$buffer;
    Initialize output$buffer;
    Initialize data$buffer;
    Initialize report$buffer;
    Initialize to immediate$mode;
    Initialize time;
End Zero$flags
```

```
                    /* Data Movement Procedures */


Send$report:
    Output ^B when ready;
    Do while report$buffer not empty
        (Output next$character when ready);
    Output ^D when ready;
End Send$report;


                    /* Interpolation Procedures */


Interpolate$next$azel:
    Add deltas to old az/el;
    Add offsets to old az/el;
    Adjust 0-360 boundary in azimuth;
    Clamp elevation to [-2,90];
    If autorun$on output$azel;
End Interpolate$next$azel;
```

```
                    /* Command Decode and Processing */


    Process$immediate$mode$record:
        Process$record;
        Decrement input$buffer record count;
        Update output$buffer;
    End Process$immediate$mode record;



    Process$autorun$record:
        Process$record;
        Update report$buffer;
        Update data$buffer;
        Decrement autorun$record$count;
    End Process$autorun record;



    Setup$autorun:
        Process$record;
        Cancel old reports;
        Initialize az/el offsets and deltas;
        Process$record;
        Update report$buffer;
        Update data$buffer;
        Set state to autorun$setup;
    End Setup$autorun;
```

```
Process$record:
    Zero digit$buffer:
    Get next$character:
    Do while next$character <> ':';
        If next$character =[numeric,sign,decimal pt.]
        then load$digit$buffer:
        Else
            (If next$character is a letter or'$' then
            set the command$code):
        If a command was detected then
            (If next$character is a demarcator then
            cndcmd:
            If abort$request then exit);
        If abort$request then exit:
        Get next$character;
    End:
    If a command was detected then
        (Perform Cndcmd: If abort$request then exit);
End Process$record


Cndcmd:
    /*Allows perform$command on certain conditions*/
    Decode digit$buffer:
    If immediate$mode command was issued then
        (If autorun$active then
            (If command is request then honor request;
            Else
                (If immediate$mode on then exclude
                set$interval:
                Else allow offsets,abort,
                set$immediate$mode));
        Else perform$command:);
    Else autorun$command was issued then
        (If autorun$on exclude set$immediate$mode:
        Else perform A,E,I,O commands);
End Cndcmd;
```

THREADER MICROCOMPUTER ANTENNA CONTROLLER
Appendix D Antenna Controller Flowgram

```
Perform$command:
    /* A brief description of commands is given.  Those
    Commands not explained are self explanatory */
    Do case command$number;
        1:   Azimuth:
            (If immediate$mode position azimuth;
            Else
                (Compute azimuth$delta; Form next azimuth
                output));
        2:   Timeoffset:
            (Advance or retard autorun track);
        3:   Elevation:
            (If immediate$mode position elevation;
            Else
                (Compute elevation$delta; Form next
                elevation output));
        4:   Set$interval:
        5:   Abort$autorun:
            (Initialize data$buffer, report$buffer;
            Turn off autorun;
            Clear pending autorun$records);
        6:   Set$immediate$mode:
        7:   ; 8:   ;   /*unused commands*/
        9:   Set$start$time:
            (Set track start time; activate autorun$mode);
        a:   ;
        b:   Set$wait$mode:
            (If numeric=0 then clear wait$mode;
            Else set$wait$mode);
        c:   ;
        d:   Autorun$done:
            (Turn off autorun;
            Process pending autorun$records);
        e:   Set azimuth$offset:
        f:   Set elevation$offset:
        10:  Request azimuth:
        11:  Request time$offset:
        12:  Request elevation:
        13:  Request interval:
        14:  ; 15:  ;   /*unused requests*/
        16:  Request report:
        17:  ;
        18:  Request real$time:
        19:  ; 1a:  ; 1b:  ; 1c:  ;
        1d:  Request azimuth$offset:
        1e:  Request elevation$offset:
        1f:  Request date:
    End;
End Perform$command;
```

```
                    /* Interrupt Processing */


Tick:
    If second$tick then
        (If countdown$on then
            (If time$now=start$time then
                (start autorun; end countdown));
        If time$request pending then transfer$time);
    If autorun$active then
        (If time$delay <> 0 then decrement time$delay;
        Else
            (If intrvl$ctr=0 then
                (set intrvlctr;
                If data$buffer not empty then up
                next$record$sem);
            Else
                (Decrement intrvlctr; up interpolation
                count));
    Else
        (If autorun$on and no countdown then begin
        countdown);
End Tick;


Pdp$sbc$isr:
    Get next$character;
    If next$character is ^X then exit to monitor;
    If input$buffer overflow imminent then
        (output ^S; set overflow flag);
    Process$character;
End Pdp$sbc$isr;


Process$character:
    If data$buffer filling then
        (If ^D then stop filling data$buffer;
        If got record$marker then update data$buffer
        record count; Update data$buffer);
    Else input$buffer being filled
        (If ^B then (enter data$buffer fill mode; exit);
        If record marker then update input$buffer record
        count; Update input$buffer);
End Process$character
```

THREADER MICROCOMPUTER ANTENNA CONTROLLER

APPENDIX E

PL/M Source Listing[Ref. 10]

```
/*****************************************************************/
/*****************************************************************/
/*                                                               */
/*    T H R E A D E R   A N T E N N A   C O N T R O L L E R      */
/*                                                               */
/*                                                               */
/*                VERSION:  1.0                                  */
/*                DATE:     7 AUGUST 1978                        */
/*                AUTHOR:   L. E. RUSSO                          */
/*                          NAVAL RESEARCH LAB                   */
/*                          CODE 7574                            */
/*                          WASHINGTON, D. C. 20375              */
/*                                                               */
/*                                                               */
/*****************************************************************/
/*****************************************************************/
/*                                                               */
/************************ GLOBAL VARIABLES ***********************/
/*                                                               */
DECLARE DCL LITERALLY 'DECLARE';
DCL LIT LITERALLY 'LITERALLY';
/*  CIRCULAR BUFFERS AND CONTROL */
DCL (OUTNDPTR,INNDPTR) BYTE;
DCL (OUTSTPTR,INSTPTR,OUTSEM,INSEM,OUTFF,INFF) BYTE;
DCL (MMIN,MMOUT) (256) BYTE;
/* AUTOMODE BUFFERS AND CONTROL */
DCL STXFLG BYTE;
DCL (DATASTPTR,DATANDPTR,RPRTBUFPTR) ADDRESS;
DCL (DATARECTR,RPRTRECTR) BYTE;
DCL DATABUF (2048) BYTE;
DCL RPRTBUF (2048) BYTE;
/*  CONTROL VARIABLES  */
DCL (CMDREG,CMDFLG,RQSTFLG) BYTE;
DCL (CNTDWNFLG,NOTSETUPFLG,AUTOFLG,STFLG,DONEFLG,TURNONFLG) BYTE;
DCL (TMRQFLG,PRIORITY,TICKSEM,NXTRECSEM) BYTE;
DCL (CTLSFLG,WAITFLG,MMOUTBUSY,ENDOUTFLG,PDPBSY,RPRTFLG) BYTE;
DCL (OLDSEC,NEWSEC) BYTE;
DCL (INTRVL,INTRVLCTR) ADDRESS;
/*  START TIME BUFFER  AND TIME OFFSET*/
DCL TSTART(3) BYTE;
DCL TMDLY ADDRESS;
/*  AZ-EL SAVE VARIABLES  */
DCL (AZHOLD,AZINC,AZOFF,ELHOLD,ELINC,ELOFF) (3) BYTE;
DCL (AZOLD,ELOLD) ADDRESS;
/*  NUMBER PROCESSING VARIABLES  */
DCL (DGTBUFPTR) BYTE;
DCL VAL ADDRESS;
DCL DGTBUF (8) BYTE;
DCL BCDASC DATA ('0','1','2','3','4','5','6','7','8'
,'9','.','-');
DCL PTR BYTE;  /*TEMPORARY POINTER*/
DCL HOLD (10) BYTE;  /*TEMPORARY BUFFER */
```

```
DCL ABRTFLG BYTE;
DCL CHAR BYTE;  /*GLOBAL CHARACTER STORAGE*/
/*                                                              */
/************************ GLOBAL LITERAL STRINGS ************************/
/*                                                              */
DCL OUT LIT 'OUTPUT';
DCL IN  LIT 'INPUT';
DCL TRUE  LIT '0FFH';
DCL FALSE LIT '0';
DCL MMFLG LIT 'NOT AUTOFLG';
DCL CHAR$IS$DIGIT LIT 'CHAR<=''9'' AND CHAR>=''0''';
DCL CTRMD LIT '0DFH';
DCL WAIT$ON$TXRDY LIT 'DO WHILE (IN(RS232$CTL) AND 1)=0; END';
    DCL RELOAD$CTR LIT 'OUT(0DFH)=70H; OUT(0DDH)=9; OUT(0DDH)=0';
/*  SERIAL IO PORTS  */
DCL RS232$CTL LIT '0EFH';
DCL RS232     LIT '0EEH';
/*  TIME CODE PORTS  */
DCL HR  LIT '0E6H';
DCL MIN LIT '0E5H';
DCL SEC LIT '0E4H';
DCL DAYLO  LIT '0E9H';
DCL DAYHI  LIT '0E9H';
/*  AZIMUTH PORTS  */
DCL AZIN1  LIT '0C4H';
DCL AZIN2  LIT '0C5H';
DCL AZOUT1 LIT '0C8H';
DCL AZOUT2 LIT '0C9H';
DCL AZOUT3 LIT '0CAH';
/*  ELEVATION PORTS  */
DCL ELIN1  LIT '0B0H';
DCL ELIN2  LIT '0B1H';
DCL ELOUT1 LIT '0B4H';
DCL ELOUT2 LIT '0B5H';
DCL ELOUT3 LIT '0B6H';
/*  SPECIAL CHARACTERS  */
DCL CTLA LIT '1';          /*START OF AUTORUN SIGNAL TO PDP11   */
DCL CTLQ LIT '21Q';        /*TELL PDP11 TO RESTART TRANSMISSION */
DCL CTLS LIT '23Q';        /*TELL PDP11 TO STOP TRANSMISSION    */
DCL CTLZ LIT '1AH';        /*SIGNAL PDP11 AUTORUN COMPLETE      */
```

```
/*                                                                    */
/********************* BEGIN UTILITY ROUTINES ************************/
/*                                                                    */
ASCBCD:         PROCEDURE (CHAR) BYTE;
    DCL CHAR BYTE;
    IF CHAR$IS$DIGIT THEN RETURN 0FH AND CHAR;
    ELSE RETURN -1;
END ASCBCD;
BCDBYTASC:          PROCEDURE(BUFAD,PTRAD,VAL);
    /*
    TAKES 2-BCD DIGIT BYTE AND RETURNS TWO ASCII DIGITS AT
    BUF(PTR),BUF(PTR+1) AND POINTS TO NEXT BUF LOCATION
    BCDBYTASC USES:
    */
    DCL (BUFAD,PTRAD) ADDRESS;
    DCL BUF BASED BUFAD  (1) BYTE;
    DCL (VAL,TEMP) BYTE;
    DCL PTR BASED PTRAD BYTE;
    TEMP=SHR(VAL,4) ;
    BUF(PTR)=BCDASC(TEMP);
    PTR=PTR+1;
    TEMP=VAL AND 0FH;
    BUF(PTR)=BCDASC(TEMP);
    PTR=PTR+1;
END BCDBYTASC;
BCDTENTHSASC:           PROCEDURE(BUFAD,PTRAD,VAL);
    /*
    CONVERT ADDRESS OF 4 BCD DIGITS TO STRING OF
      OF ASCII CHARACTERS, HUNDREDS THROUGH TENTHS

    BCDTENTHSASC USES:  LOW,HIGH,SHR
    */
    DCL (PTRAD,BUFAD,VAL) ADDRESS;
    DCL BUF BASED BUFAD (1) BYTE;
    DCL PTR BASED PTRAD ADDRESS;
    DCL (I,HVAL,LVAL) BYTE;
    DCL IF$SIGN$IS$ON LIT 'I=HIGH(VAL)+0; IF SIGN';
    DCL LOAD$BUF LIT 'BUF(PTR)=BCDASC(I); PTR=PTR+1';
    IF$SIGN$IS$ON THEN
        DO;
            I=11;     /*WRITE MINUS SIGN*/
            LOAD$BUF;
        END;
    I=SHR(HVAL:=HIGH(VAL),4) AND 07H;
    LOAD$BUF;     /*WRITE ASCII HUNDREDS*/
    I=HVAL AND 0FH;     /*WRITE ASCII TENS*/
    LOAD$BUF;
    I=SHR(LVAL:=LOW(VAL),4) ;     /*WRITE ASCII ONES*/
    LOAD$BUF;
    I=10;     /*WRITE DECIMAL POINT*/
    LOAD$BUF;
    I= LVAL AND 0FH;
    LOAD$BUF;     /*WRITE ASCII TENTHS*/
END BCDTENTHSASC;
```

```
OUT$STRNG:          PROCEDURE (VAL,INTERP,CHAR);
    /*
    WRITES ASCII STRING TO APPROPRIATE BUFFER

    OUT$STRNG USES:  BCDTENTHSASC
    */
    DCL (CHAR,INTERP,I) BYTE;
    DCL (VAL,PTR) ADDRESS;
    IF INTERP=0 THEN
        DO;
            HOLD(PTR:=0)=CHAR;
            PTR=PTR+1;
            CALL BCDTENTHSASC(.HOLD,.PTR,VAL);
            DO I=0 TO (PTR-1);
                MMOUT(OUTNDPTR)=HOLD(I);
                OUTNDPTR=OUTNDPTR+1;
                IF ZERO THEN OUTFF=TRUE;  /*CHECK BUFFER OVERFLOW? */
            END;
        END;
    ELSE
        DO;
            RPRTBUF(RPRTBUFPTR)=CHAR;
            RPRTBUFPTR=RPRTBUFPTR+1;
            CALL BCDTENTHSASC(.RPRTBUF,.RPRTBUFPTR,VAL);
        END;
END OUT$STRNG;
DECODE:                 PROCEDURE (BUFAD,BUFPTR) ADDRESS;
    /*
    DECODE PROCESSES AN ASCII STRING '  DDD.D' AND RETURNS
    A BCD NUMBER IN TENTHS

    DECODE USES:  ASCBCD
    */
    DCL (TENTHS,BUFAD,BUFPTR,TEMP) ADDRESS;
    DCL BUF BASED BUFAD (1) BYTE;
    DCL ENDBUF BASED BUFPTR BYTE;
    DCL (I,K,CHAR) BYTE;
    DCL K$IS$POS LIT '(K AND 80H)=0';
    CHAR=BUF(K:=0);
    DO WHILE CHAR<> '.' AND K<ENDBUF;
        CHAR=BUF(K:=K+1);
    END;
    IF K<>ENDBUF THEN TENTHS=ASCBCD(BUF(K+1));
    ELSE TENTHS=0;
    I=0;
    DO WHILE K<>0;
        CHAR=BUF(K:=K-1);
        IF CHAR$IS$DIGIT THEN
            DO;
                I=I+4;
                TEMP=ASCBCD(CHAR);
                TENTHS=SHL(TEMP,I) + TENTHS;
            END;
```

```
        ELSE
            DO;
                IF CHAR='-' THEN
                    DO;
                        TENTHS=TENTHS OR 8000H;
                        K=0;
                    END;
            END;
    END;
    RETURN TENTHS;
END DECODE;
BCDBIN:                 PROCEDURE (BCD) ADDRESS;
    /*
    CONVERTS 4 BCD DIGITS TO BINARY,
    SIGNED MAGNITUDE IS ASSUMED: THEREFORE, +-799.9
    IS RANGE OF BCD

    BCDBIN USES:  SHL,SHR,LOW,HIGH
    */
    DCL (BCD,BIN,VAL) ADDRESS;
    DCL (TEMP,SGNFLG) BYTE;
    IF (HIGH(BCD) AND 80H)<>0 THEN
        DO;
            SGNFLG=TRUE;
            BCD=BCD AND 7FFFH;
        END;
    ELSE SGNFLG=FALSE;
    /*  CONVERT MAGNITUDE TO BINARY  */
    VAL=SHR(TEMP:=HIGH(BCD),4);
    BIN=SHL(VAL,3);
    BIN=BIN + SHL(VAL,1);  /* TIMES 10*/
    TEMP=TEMP AND 0FH;
    VAL=TEMP+BIN;
    BIN=SHL(VAL,3);
    BIN=SHL(VAL,1) +BIN;  /* TIMES 10 AGAIN */
    VAL=SHR(TEMP:=LOW(BCD),4)+BIN;
    BIN=SHL(VAL,3);
    BIN=SHL(VAL,1) + BIN; /*TIMES 10 AGAIN*/
    TEMP=TEMP AND 0FH;
    BIN=TEMP + BIN;
    /*  FORM TWO'S COMPLEMENT BINARY  */
    IF SGNFLG THEN RETURN -BIN;
    ELSE RETURN BIN;
END BCDBIN;
QUOREM:         PROCEDURE (REMPTR,TENXXN) BYTE;
    /*
    QUOREM RETURNS THE QUOTIENT OF REM/TENXXN WHEN
    TENXXN IS A POWER OF TEN.  REM IS MODIFIED TO YIELD
    THE REMAINDER.  REM IS ASSUMED TO BE A POSITIVE INTEGER.

    QUOREM USES:
    */
```

```
    DCL (REMPTR,TENXXN) ADDRESS;
    DCL REM BASED REMPTR ADDRESS;
    DCL QUO BYTE;
    QUO=0;
    DO WHILE REM>=TENXXN AND QUO<10;
        QUO=QUO+1;
        REM=REM-TENXXN;
    END;
    RETURN QUO;
END QUOREM;
TENSCOMP:       PROCEDURE (BCD) ADDRESS;
    /*
    TENSCOMP RRETURNS THE TEN'S COMPLEMENT OF A POSITIVE MAGNITUDE
    BCD NUMBER IN BCD FORMAT

    TENSCOMP USES:  DEC,HIGH, LOW
    */
    DCL (BCD,BCDPTR) ADDRESS;
    DCL BCDB BASED BCDPTR (2) BYTE;
    BCDPTR=.BCD;
    BCD=9999H-BCD;
    BCDB(0)=DEC(LOW(BCD)+1);
    BCDB(1)=DEC(HIGH(BCD) PLUS 0);
    RETURN BCD;
END TENSCOMP;
BINBCD:         PROCEDURE(BIN) ADDRESS;
    /*
    CONVERT BINARY ADDRESS TO BCD COUNTERPART. IF BIN>7999,
    THEN 7999H IS RETURNED.

    BINBCD USES:  MOD./
    */
    DCL (QUOT,REM,BIN) ADDRESS;
    DCL SGNFLG BYTE;
    IF (HIGH(BIN) AND 80H) <> 0 THEN
        DO;
            REM=-BIN;
            SGNFLG=TRUE;
        END;
    ELSE
        DO;
            REM=BIN;
            SGNFLG=FALSE;
        END;
    QUOT=QUOREM(.REM,1000);
    IF QUOT>7 THEN
        DO;
            IF SGNFLG THEN RETURN 0F999H;
            ELSE RETURN 7999H;
        END;
```

```
    QUOT=QUOREM(.REM,100) + SHL(QUOT,4);
    QUOT=QUOREM(.REM,10) + SHL(QUOT,4);
    QUOT=REM + SHL(QUOT,4);
    IF SGNFLG THEN RETURN QUOT OR 8000H;
    ELSE RETURN QUOT;
END BINBCD;
BCDADD:         PROCEDURE(BCD1,BCD2) ADDRESS;
    /*
    BCDADD ADDS TWO 4-DIGIT BCD, SIGN-MAGNITUDE NUMBERS
    AND RETURNS THE 4-DIGIT SIGN MAGNITUDE RESULT.
    BCDADD USES:  HIGH,LOW,DEC
    */
    DCL (BCD1,BCD2,SUMPTR) ADDRESS;
    DCL SGNSEM BYTE;
    DCL SUMB(2) BYTE;
    DCL SUM BASED SUMPTR ADDRESS;
    SUMPTR=.SUMB;
    SGNSEM=0;
    SUMB(0)=HIGH(BCD1) + 0;
    IF SIGN THEN
        DO;
            BCD1=TENSCOMP(BCD1 AND 7FFFH);
            SGNSEM=SGNSEM+1;
        END;
    SUMB(1)=HIGH(BCD2) + 0;
    IF SIGN THEN
        DO;
            BCD2=TENSCOMP(BCD2 AND 7FFFH);
            SGNSEM=SGNSEM+1;
        END;
    SUMB(0)=DEC(LOW(BCD1) + LOW(BCD2));
    SUMB(1)=DEC(HIGH(BCD1) PLUS HIGH(BCD2));
    IF SIGN OR (SGNSEM=2) THEN RETURN TENSCOMP(SUM) OR 8000H;
    ELSE RETURN SUM;
END BCDADD;
TENSCOMP3:          PROCEDURE(BCDPTR);
    /*
    COMPUTES TENSCOMP OF 3-BYTE BCD STRING

    TENSCOMP3 USES:  DEC
    */
    DCL BCDPTR ADDRESS;
    DCL BCD BASED BCDPTR (3) BYTE;
    DCL BCD2 BYTE;
    BCD(0)=99H-BCD(0);
    BCD(1)=99H-BCD(1);
    BCD2=99H-BCD(2);
    BCD(0)=DEC(BCD(0) + 1);
    BCD(1)=DEC(BCD(1) PLUS 0);
    BCD2=DEC(BCD2 PLUS 0);
    BCD(2)=BCD2;
END TENSCOMP3;
```

```
BCDADD3:        PROCEDURE (BCD1PTR,BCD2PTR,BCD3PTR);
    /* 3 BYTE BCD ADD: BCD3=BCD1+BCD2. 10'S COMPLEMENT
    INPUT ASSUMED

    BCDADD3 USES:  DEC
    */
    DCL (BCD1PTR,BCD2PTR,BCD3PTR) ADDRESS;
    DCL BCD1 BASED BCD1PTR (3) BYTE;
    DCL BCD2 BASED BCD2PTR (3) BYTE;
    DCL BCD3 BASED BCD3PTR (3) BYTE;
    DCL (BCD12,BCD22,BCD32) BYTE;
    BCD12=BCD1(2);
    BCD22=BCD2(2);
    BCD3(0)=DEC(BCD1(0) + BCD2(0));
    BCD3(1)=DEC(BCD1(1) PLUS BCD2(1));
    /* CODE CORRECTS COMPILER ERROR */
    BCD32=DEC(BCD12 PLUS BCD22);
    BCD3(2)=BCD32;
END BCDADD3;
OUT$CHAR:       PROCEDURE       (CHAR);
    /*
    OUT$CHAR ENABLES OUTPUT OF A SINGLE CHARACTER WITHOUT
    OVERWRITING THE OUTPUT BUFFER.
    THIS ROUTINE IS USED FOR SENDING CONTROL CHARACTERS
    TO THE PDP11.

    OUT$CHAR USES:
    */
    DCL CHAR BYTE;
    WAIT$ON$TXRDY;
    OUT(RS232)=CHAR;
END OUT$CHAR;
/********************* END UTILITY ROUTINES **********************/
```

```
/*                                                                    */
/****************** BEGIN PROCESSOR SET-UP ROUTINES ******************/
/*                                                                    */
ZERO$FLGS:          PROCEDURE;
    /*
    INITIALIZE GLOBAL VARIABLES
    */
    /* INITIALIZE INPUT BUFFER */
    INNDPTR=0;
    INSEM=0;
    INFF=FALSE;
    INSTPTR=0;
    /* INITIALIZE OUTPUT BUFFER */
    OUTNDPTR=0;
    OUTSEM=0;
    OUTFF=FALSE;
    OUTSTPTR=0;
    /* INITIALIZE REPORT BUFFER */
    RPRTBUFPTR=0;
    RPRTFLG=FALSE;
    RPRTRECTR=0;
    /* INITIALIZE DATA BUFFER */
    STXFLG=FALSE;
    DATASTPTR=0;
    DATANDPTR=0;
    DATARECTR=0;
    /* RESET AUTORUN CONTROL */
    NXTRECSEM=0;
    AUTOFLG=FALSE;
    CNTDWNFLG=FALSE;
    STFLG=FALSE;
    INTRVL=10;  /*INTRVL DEFAULTS TO 1.0 SEC*/
    TMDLY=0;
    NOTSETUPFLG=FALSE;
    ABRTFLG=FALSE;
    TICKSEM=0;
    /* TURN OFF CHARACTER OUTPUT */
    ENDOUTFLG=FALSE;
    MMOUTBUSY=FALSE;
    /* INITIALIZE SERIAL CHANNEL CONTROL */
    PDPBSY=FALSE;
    CTLSFLG=FALSE;
    /* RESET COMMAND PROCESSING */
    RQSTFLG=FALSE;
    TMRQFLG=FALSE;
    WAITFLG=FALSE;
    DGTBUFPTR=0;
    /* INITIALIZE EVENT CLOCK CONTROL */
    NEWSEC=0;
    OLDSEC=0;
END ZERO$FLGS;
```

```
SET$IO:         PROCEDURE;
    /*SET UP CTR1 FOR TIME CODE INTERRUPT EVERY .1 SEC*/
    RELOAD$CTR;
    /* SET UP PARALLEL IO FOR SBC80/20 */
    DCL AZ1CTL LIT '0C7H';
    DCL AZ2CTL LIT '0C8H';
    DCL EL1CTL LIT '0B3H';
    DCL EL2CTL LIT '0B7H';
    OUT(AZ1CTL)=90H;  /*INPUT ON SBC108 GRP 1 */
    OUT(AZ2CTL)=80H;  /*OUTPUT ON SBC108 GRP 2*/
    OUT(EL1CTL)=90H;  /*INPUT ON SBC519 GRP 1 */
    OUT(EL2CTL)=80H;  /*OUTPUT ON SBC519 GRP 2*/
    /*SBC80/20 PORTS DEFAULT TO INPUT*/
    /*SET AZOUT3, ELOUT3 PORTS SO THAT UPDATE IS DISABLED
    AND PORTS HAVE ZERO OUTPUT AFTER DRIVERS*/
    OUT(AZOUT3)=NOT 0;
    OUT(ELOUT3)=NOT 0;
END SET$IO;
SET$INT:        PROCEDURE;
    /*
    SET-UP INTERRUPT TABLES FOR 8259 PRIORITY INTERRUPT CHIP
    SET-UP COMPATIBLE WITH SBC 80/20 MONITOR.  SEE
    SBC80P20 USERS GUIDE,P. 19.

              LEVEL          INTERRUPT
                0        PROCESSING/RESTART
                1        RETURN TO 80/20 MONITOR.
                2        80/20 SINGLE STEP(TIMER)
                3        .1 SEC. TICK
                4        PDP TO SBC COMM.
                5        SBC TO PDP COMM.
                6        SPARE

    CODE IS NOT VALID IN PLM BUT SHOWS WHAT WE WANT TO DO:
    DCL INT$VECT$ST ADDRESS;
    DCL INT$VECT BASED INT$VECT$ST (16) ADDRESS;
    INT$VECT$ST=3FE0H:        !DEFINE TABLE START. ADDRESS
    INT$VECT(10)=.SBC$PDP$ISR;  INTERRUPT VECTOR 5
    INT$VECT(8)=.PDP$SBC$ISR;  INTERRUPT VECTOR 4
    INT$VECT(6)=.TICK;        INTERRUPT VECTOR 3
    */
    DCL INT$ST07$OFF LIT '0E0H';
    DCL OCW1 LIT '0D9H';
    DCL INT$SETUP LIT '3800H';
    OUT(OCW1)=INT$ST07$OFF;
    GO TO INT$SETUP;          /*LINK TO ASSEMBLY LANGUAGE */
    ENABLE;
END SET$INT;
```

```
SET$TIM:        PROCEDURE;
    /* FORMS BCD TIME FROM ASCII STRING.  NO
    NON-DIGIT CHARACTERS ARE ALLOWED.  TSTART ARRAY WILL
    CONTAIN BCD VALUES FOR 0-23 HRS., 0-59 MINS.
    AND 0-59 SECS.  IF LESS THAN 6 DIGITS ARE PRESENT,
    ZEROES ARE INSERTED FOR REMAINING CHARACTERS.
    SET$TIM USES:  ASCBCD  */
    DCL (TVAL,I,PTR) BYTE;
    DO PTR=DGTBUFPTR TO 5;
        DGTBUF(PTR)=0;
    END;
    I=-1;
    DO PTR=0 TO 4 BY 2;
        TVAL=SHL(ASCBCD(DGTBUF(PTR)),4);
        TSTART(I:=I+1)=TVAL + ASCBCD(DGTBUF(PTR+1));
    END;
    AUTOFLG=TRUE;  /*SETTING TIME COMMENCES AUTO-MODE OPERATION*/
END SET$TIM;
/********************* END PROCESSOR SET UP ROUTINES **********************/
```

```
/*                                                                    */
/****************** BEGIN DATA MOVEMENT ROUTINES ********************/
/*                                                                    */
OUT$REC:        PROCEDURE;
    /*
    TRIGGERS  OUTPUT OF 1 RECORD.
    METHOD IS BASED ON STATUS OF XMIT. BUF. RDY. AND CARE
    IS TAKEN TO ENSURE NO POSSIBILITY OF OVERWRITING A
    CURRENT OUTPUT

    OUT$REC USES:
    OUT$REC IS CALLED BY:
    */
    DCL CHAR BYTE;
    WAIT$ON$TXRDY;
    OUT(RS232)=(CHAR:=MMOUT(OUTSTPTR));
    IF CHAR=':' THEN
        DO;
            OUTSEM=OUTSEM-1;
            ENDOUTFLG=TRUE;
        END;
    OUTSTPTR=OUTSTPTR+1;
    IF ZERO THEN OUTFF=FALSE;
    MMOUTBUSY=TRUE;
END OUT$REC;
TIM$TO$BUF:       PROCEDURE(BUFAD,PTRAD);
    DCL (BUFAD,PTRAD) ADDRESS;
    DCL BUF BASED BUFAD (1) BYTE;
    DCL PTR BASED PTRAD BYTE;
    BUF(PTR)='T';
    PTR=PTR+1;
    CALL BCDBYTASC(BUFAD,PTRAD,IN(HR));
    CALL BCDBYTASC(BUFAD,PTRAD,IN(MIN));
    CALL BCDBYTASC(BUFAD,PTRAD, NOT (IN(SEC)));
END TIM$TO$BUF;
OUTPUT$AZIM:        PROCEDURE(TEMPTR);
    /*
    OUTPUT$AZIM OUTPUTS AZIMUTH VALUE TO HUNDREDTHS
    TO THE ANTENNA SYSTEM.
    UPDATE MODE IS TURNED ON/OFF FOR OUTPUT.

    OUTPUT$AZIM USES:  SHR,DEC
    */
    DCL TEMPTR ADDRESS;
    DCL TEMP BASED TEMPTR (3) BYTE;
    DCL HUND BYTE;
    DISABLE;
    /*ROUND HUNDREDTHS AND SHIFT TO LOW NIBBLE*/
    HUND=SHR(DEC(TEMP(0) + 5),4);
    OUT(AZOUT3) = NOT (HUND OR 80H);  /*UPDATE MODE ON*/
    OUT(AZOUT1) = NOT TEMP(1);
    OUT(AZOUT2) = NOT TEMP(2);
    OUT(AZOUT3) = NOT HUND;           /*UPDATE MODE OFF*/
    ENABLE;
END OUTPUT$AZIM;
```

```
OUTPUT$ELEV:        PROCEDURE(TEMPTR);
    /*
    OUTPUT$ELEV OUTPUTS ELEVATION VALUE TO HUNDREDTHS
    TO THE ANTENNA SYSTEM.
    UPDATE MODE IS TURNED ON/OFF FOR OUTPUT.

    OUTPUT$ELEV USES:  SHR,DEC
    */
    DCL TEMPTR ADDRESS;
    DCL TEMP BASED TEMPTR (3) BYTE;
    DCL HUND BYTE;
    DISABLE;
    /*ROUND HUNDREDTHS AND SHIFT TO LOW NIBBLE*/
    HUND=SHR(DEC(TEMP(0) + 5),4);
    OUT(ELOUT3) = NOT (HUND OR 80H);  /*UPDATE MODE ON*/
    OUT(ELOUT1) = NOT TEMP(1);
    OUT(ELOUT2) = NOT TEMP(2);
    OUT(ELOUT3) = NOT HUND;           /*UPDATE MODE OFF*/
    ENABLE;
END OUTPUT$ELEV;
REPORT:                 PROCEDURE;
    /*        SENDS CONTENTS OF REPORT TO PDP11 AS A
     DATA FILE, IE. FRAMED BY CTLB...CTLD.  ONCE ACTIVATED, THE
     REPORT CONTINUES UNTIL THE WHOLE BUFFER IS
     TRANSFERRED.  TRANSFERRING 2048 CHARACTERS AT 2400 BD.
     WILL TAKE ABOUT 10 SEC.  IF A NEW RUN IS ACTIVATED
     BEFORE THE REPORT IS MADE, THE REQUEST FOR THE
     REPORT IS CANCELLED.

     REPORT USES:
     REPORT IS CALLED BY: PRIORITY RESOLVER
    */
    DCL PTR ADDRESS;
    WAIT$ON$TXRDY;
    OUT(RS232)=02;          /*OUTPUT CTLB*/
    PTR=0;
    DO WHILE RPRTRECTR>0;
        CHAR=RPRTBUF(PTR);
        PTR=PTR+1;
        IF CHAR=':' THEN RPRTRECTR=RPRTRECTR-1;
        WAIT$ON$TXRDY;
        OUT(RS232)=CHAR;
    END;
    WAIT$ON$TXRDY;
    OUT(RS232)=04;          /*OUTPUT CTLD*/
    RPRTFLG=FALSE;
END REPORT;
```

```
POSITION:       PROCEDURE(AZEL,AZON);
    /*
    POSITION POSITIONS ANTENNA TO AZ/EL POSITION.  IF
    AZON=1 THEN POSITION AZIMUTH ELSE ELEVATION.  IF WAITFLG
    THEN DO NOT RETURN UNTIL ANTENNA IS WITHIN
    .5 DEG. OF POSITION

    POSITION USES:  LOW,HIGH,TENSCOMP,BCDADD
    */
    DCL (AZEL,TEMPTR,TEST,TEMP) ADDRESS;
    DCL AZON BYTE;
    DCL TEMPB BASED TEMPTR (2) BYTE;
    DCL AZELARY (3) BYTE;
    /*ZERO HUNDREDTHS*/
    AZELARY(0)=0;
    /*DEFINE TENTHS THRU HUNDREDS*/
    AZELARY(1)=LOW(AZEL);
    AZELARY(2)=HIGH(AZEL);
    TEMPTR=.TEMP;
    IF AZON THEN CALL OUTPUT$AZIM(.AZELARY);
    ELSE CALL OUTPUT$ELEV(.AZELARY);
    IF WAITFLG THEN
        DO;
            TEST=10;
            DO WHILE TEST>5;
                IF AZON THEN
                    DO;
                        DISABLE;
                        TEMPB(0)= NOT IN(AZIN1);
                        TEMPB(1)=IN(AZIN2) AND 0BFH;
                        ENABLE;
                    END;
                ELSE
                    DO;
                        DISABLE;
                        TEMPB(0)= NOT IN(ELIN1);
                        TEMPB(1)=IN(ELIN2) AND 0BFH;
                        ENABLE;
                    END;
                TEMP=TENSCOMP(TEMP);
                TEST=BCDADD(TEMP,AZEL) AND 7FFFH;  /* GET MAGNITUDE */
            END;
        END;
END POSITION;
```

```
TM$OFFSET:                  PROCEDURE;
    /*
    MODIFY AUTORUN TO ADVANCE OR RETARD SATELLITE
    TRACK WITH RESPECT TO REAL TIME EPHEMEROUS DATA

    TM$OFFSET USES:
    TM$OFFSET IS CALLED BY: PRIORITY RESOLVER
    */
    DCL (NSTPS,NRCDS,BINOFF,TICKDIF) ADDRESS;
    BINOFF=BCDBIN(VAL AND 7FFFH);
    IF (HIGH(VAL) AND 80H)<>0 THEN
        DO:                                 /*ADVANCE TRACK TIME */
            NRCDS=BINOFF/INTRVL;
            NSTPS=BINOFF MOD INTRVL;
            DISABLE;
            TICKDIF=INTRVLCTR-NSTPS;
            IF NSTPS>=INTRVLCTR THEN
                DO;
                    INTRVLCTR=INTRVL+TICKDIF;
                    TICKSEM=-TICKDIF-1;
                    NXTRECSEM=NRCDS+NXTRECSEM+1;
                END;
            ELSE
                DO;
                    INTRVLCTR=TICKDIF;
                    NXTRECSEM=NRCDS+NXTRECSEM;
                    TICKSEM=INTRVL-TICKDIF-1;
                END;
            ENABLE;
            TMDLY=0;                        /*ADVANCE GTR THAN REAL TRACK TIME*/
        END;
    ELSE TMDLY=BINOFF;                      /*TRUE TRACK TIME DELAY */
END TM$OFFSET;
/************** END DATA MOVEMENT ROUTINES **************************/
```

```
/*                                                                    */
/********************* INTERPOLATION ROUTINES ************************/
/*                                                                    */
ADJUST:         PROCEDURE         (HOLDPTR);
    /*
    THIS ROUTINE ADJUSTS FOR 359.9 TO 0.0 AND 0.0 TO 359.9
    TRANSITIONS SO THAT THE OUTPUT WILL BE KEPT IN THE
    INTERVAL [0,359.9].

    ADJUST USES:  DEC
    */
    DCL HOLDPTR ADDRESS;
    DCL HOLD BASED HOLDPTR (3) BYTE;
    IF HOLD(2)>49H THEN HOLD(2)=DEC(HOLD(2) + 36H);
    ELSE
        DO;
            IF HOLD(2)>35H THEN HOLD(2)=DEC(HOLD(2) + 64H);
        END;
END ADJUST;
CLAMP:          PROCEDURE         (ELEV)          ADDRESS;
    /*
    CLAMP CLAMPS THE ELEVATION OUTPUT IN THE RANGE
    -2 DEG. TO 90. DEG.  ELEV IS A TENSCOMP NUMBER; THE
    RETURN IS A BCD SIGNED-MAGNITUDE NUMBER

    CLAMP USES:  TENSCOMP
    */
    DCL ELEV ADDRESS;
    IF ELEV>4999H THEN
        DO;
            ELEV=TENSCOMP(ELEV);
            IF ELEV>0020 THEN ELEV=0020H;
            ELEV=ELEV OR 8000H;
        END;
    ELSE
        DO;
            IF ELEV>0900H THEN ELEV=0900H;
        END;
    RETURN ELEV;
END CLAMP;
```

```
INC$AZEL:       PROCEDURE;
    /*
    INC$AZEL PROVIDES THE NEXT INTERPOLATED
    AZIMUTH AND ELEVATION VALUES BETWEEN RECORDS
    OF POINTING DATA.  IF A OR E IS MISSING IN A RECORD, THE
    OLD OFFSET AND INCREMENT CONTINUE TO BE USED IN THE
    INTERPOLATION.

    INC$AZEL USES:  BCDADD3, ADJUST
    */
    DCL ELEVPTR ADDRESS;
    DCL ELEV BASED ELEVPTR ADDRESS;
    DCL TEMP (3) BYTE;
    CALL BCDADD3(.AZHOLD,.AZINC,.AZHOLD);
    CALL BCDADD3(.AZHOLD,.AZOFF,.TEMP);
    CALL ADJUST(.TEMP);
    IF AUTOFLG THEN
        DO;
        /*OUTPUT ONLY IF OPERATOR HAS NOT INTERRUPTED RUN*/
            CALL OUTPUT$AZIM(.TEMP);
        END;
    CALL BCDADD3(.ELHOLD,.ELINC,.ELHOLD);
    CALL BCDADD3(.ELHOLD,.ELOFF,.TEMP);
    ELEVPTR=.TEMP(1);
    ELEV=CLAMP(ELEV);
    IF AUTOFLG THEN CALL OUTPUT$ELEV(.TEMP);
END INC$AZEL;
```

```
SET$INC:                PROCEDURE(NEW,CURRENT) ADDRESS;
    /*
    SET$INC SETS THE INCREMENT FOR THE NEW AZ-EL
    COMMAND.  THE CURRENT AZ-EL IS SUPPLIED ALONG WITH
    THE NEW AZ-EL POSITION DESIRED.  THE INCREMENT IS THE
    (NEW-CURRENT)/INTRVL.

    SET$INC USES:  BCDBIN,BINBCD,/
    */
    DCL (NEW,CURRENT,INC,DIFF,NEWBIN,CURBIN,POSDIF) ADDRESS;
    DIFF=(NEWBIN:=BCDBIN(NEW))-(CURBIN:=BCDBIN(CURRENT));
    IF NEWBIN<900 AND CURBIN>2700 THEN DIFF=DIFF + 3600;
    IF NEWBIN>2700 AND CURBIN<900 THEN DIFF=DIFF - 3600;
    IF (HIGH(DIFF) AND 80H)<>0 THEN POSDIF=-DIFF;
    ELSE POSDIF=DIFF;
    IF POSDIF>327 THEN
        DO;          /*  INCREMENT>32.7 DEG.  */
            INC=(20*POSDIF)/INTRVL;
            INC=SHR(INC+1,1);          /*ROUNDING*/
            INC=10*INC;
        END;
    ELSE
        DO;          /*  INCREMENT<=32.7 DEG.  */
            INC=(200*POSDIF)/INTRVL;
            INC=SHR(INC+1,1);
        END;
    IF(HIGH(DIFF) AND 80H)<>0 THEN INC=-INC;
    /*  INC IS ACCURATE TO .001 DEG.  */
    RETURN BINBCD(INC);
END SET$INC;
SET$AZEL$INC:          PROCEDURE(INC,AZELPTR);
    /*
    FORMS 3-BYTE 10'S COMPLEMENT INCREMENT FOR AZ OR EL

    SET$AZEL$INC USES:  LOW,HIGH
    */
    DCL (INC,AZELPTR) ADDRESS;
    DCL AZEL BASED AZELPTR (3) BYTE;
    DCL SGNFLG BYTE;
    IF (HIGH(INC) AND 80H)<>0 THEN
        DO;
            INC=TENSCOMP(INC AND 7FFFH);
            SGNFLG=TRUE;
        END;
    ELSE SGNFLG=FALSE;
    AZEL(0)=LOW(INC);
    AZEL(1)=HIGH(INC);
    IF SGNFLG THEN AZEL(2)=99H;
    ELSE AZEL(2)=0;
END SET$AZEL$INC;
/********************* END INTERPOLATION ROUTINES **********************/
```

```
/*                                                                    */
/********** COMMAND DECODE AND PROCESSING **********/
/*                                                                    */
PRFRM$CMD:        PROCEDURE(CMDCD,INTERP);
    /*
    PERFORMS ACTUAL INTERPRETATION OF COMMANDS.  31 COMMANDS
    ARE POSSIBLE:  16 REQUESTS, 15 COMMANDS

    PRFRM$CMD USES:  ZERO$FLGS,SET$TIM,OUT$STRNG
    */
    DCL (CMDCD,INTERP) BYTE;
    DCL (CURRENTPTR,INCPTR,TEMPA) ADDRESS;
    DCL CURRENT BASED CURRENTPTR ADDRESS;
    DCL INC BASED INCPTR ADDRESS;
    DCL TEMP (3) BYTE;
    IF CMDCD>1FH OR CMDCD=0FFH THEN RETURN;
    CURRENTPTR=.TEMP(1);
    TEMP(0)=0;
    DO CASE CMDCD;
        DO;                                     /*CASE AZIMUTH*/
            DO CASE INTERP;
                CALL POSITION(VAL,1);
                DO;
                    INCPTR=.AZINC(1);
                    TEMP(1)= LOW(AZOLD);
                    TEMP(2)= HIGH(AZOLD);
                    INC=SET$INC(VAL,CURRENT);
                    CALL SET$AZEL$INC(INC,.AZINC);
                    CALL BCDADD3(.AZINC,.TEMP,.AZHOLD);
                    CALL BCDADD3(.AZHOLD,.AZOFF,.TEMP);
                    CALL ADJUST(.TEMP);
                    IF AUTOFLG THEN CALL OUTPUT$AZIM(.TEMP);
                    AZOLD=VAL;
                END;
                DO;
                    CALL POSITION(VAL,1);
                    AZOLD=VAL;
                END;
            END;
        END;
        CALL TM$OFFSET;                         /*CASE TIMEOFFSET*/
        DO;                                     /*CASE ELEVATION */
            DO CASE INTERP;
                CALL POSITION(VAL,0);
                DO;
                    INCPTR=.ELINC(1);
                    TEMP(1)= LOW(ELOLD);
                    TEMP(2)= HIGH(ELOLD);
                    INC=SET$INC(VAL,CURRENT);
                    CALL SET$AZEL$INC(INC,.ELINC);
                    CALL BCDADD3(.ELINC,.TEMP,.ELHOLD);
                    CALL BCDADD3(.ELHOLD,.ELOFF,.TEMP);
                    CURRENT=CLAMP(CURRENT);
                    IF AUTOFLG THEN CALL OUTPUT$ELEV(.TEMP);
                    ELOLD=VAL;
                END;
```

```
            DO;
                ELOLD=VAL;
                CALL POSITION(VAL,0);
            END;
        END;
    END;
    INTRVL=BCDBIN(VAL);                    /*CASE SET INTERVAL */
    DO;                                    /*CASE ABORT RUN*/
        ABRTFLG=TRUE;
        STFLG=FALSE;
        AUTOFLG=FALSE;
        DATASTPTR=0;
        DATANDPTR=0;
        DATARECTR=0;
        RPRTBUFPTR=0;
        RPRTRECTR=0;
        NXTRECSEM=0;
    END;
    DO;                                    /*CASE SET MANUAL MODE*/
        IF VAL <>0 THEN AUTOFLG=FALSE;
        ELSE AUTOFLG=TRUE;
    END;
    ;;                                     /*CASE SET ANT. STAT.-TO BE ENTERED*/
    CALL SETSTIM;                          /*CASE SET TRACK START TIME*/
    ;                                      /*CASE NULL COMMAND*/
    DO;                                    /*CASE ENTER WAIT MODE*/
        IF VAL<>0 THEN WAITFLG=TRUE;
        ELSE WAITFLG=FALSE;
    END;
    ;                                      /*CASE NULL COMMAND*/
    DO;                                    /*CASE PAPER TAPE DONE*/
        AUTOFLG=FALSE;
        STFLG=FALSE;
        DONEFLG=TRUE;
        DATANDPTR=0;
        RPRTBUFPTR=0;
        /*IF NXTRECSEM<>0, PENDING RECORDS WILL STILL BE PROCESSED*/
    END;
    DO;                                    /*CASE AZIMUTH OFFSET*/
        CURRENTPTR=.AZOFF(1);
        IF (HIGH(VAL) AND 80H)=0 THEN CURRENT=VAL;
        ELSE CURRENT=TENSCOMP(VAL AND 7FFFH);
        AZOFF(0)=0;
    END;
    DO;                                    /*CASE ELEVATION OFFSET*/
        CURRENTPTR=.ELOFF(1);
        IF (HIGH(VAL) AND 80H)=0 THEN CURRENT=VAL;
        ELSE CURRENT=TENSCOMP(VAL AND 7FFFH);
        ELOFF(0)=0;
    END;
    ;                                      /*UNDEFINED COMMANDS*/
```

```
    DO:                                     /*CASE REQUEST AZIMUTH*/
        DISABLE;
        TEMP(1)= NOT IN(AZIN1);
        TEMP(2)=IN(AZIN2) AND 0BFH;
        ENABLE;
        CALL OUT$STRNG(CURRENT,INTERP,'A');
    END;
    DO:                                     /*CASE REQUEST TIME OFFSET*/
        CALL OUT$STRNG(BINBCD(TMDLY),INTERP,'O');
    END;
    DO:                                     /*CASE REQUEST ELEVATION*/
        DISABLE;
        TEMP(1)= NOT IN(ELIN1);
        TEMP(2)=IN(ELIN2) AND 0BFH;
        ENABLE;
        CALL OUT$STRNG(CURRENT,INTERP,'E');
    END;
    DO:                                     /*CASE REQUEST INTERVAL*/
        CURRENT=BINBCD(INTRVL);
        CALL OUT$STRNG(CURRENT,INTERP,'I');
    END;
    :                                         /*UNDEFINED REQUESTS*/
    DO:                                     /*CASE REQUEST IMMEDIATE MODE*/
        IF AUTOFLG THEN CALL OUT$STRNG(0,INTERP,'M');
        ELSE CALL OUT$STRNG(1,INTERP,'M');
    END;
    RPRTFLG=TRUE;                           /*CASE SEND REPORT*/
    :                                       /*UNDEFINED REQUESTS*/
    DO:                                     /*CASE REQUEST TIME*/
        IF INTERP=0 THEN TMRQFLG=TRUE;
        ELSE CALL TIM$TO$BUF(.RPRTBUF,.RPRTBUFPTR);
    END;
    :                                     /*UNDEFINED REQUESTS*/
    DO:                                     /*CASE REQUEST WAITMODE*/
        IF WAITFLG THEN CALL OUT$STRNG(1,INTERP,'W');
        ELSE CALL OUT$STRNG(0,INTERP,'W');
    END;
    ::                                /*UNDEFINED REQUESTS*/
    DO:                                     /*CASE REQUEST AZIM. OFFSET*/
        CURRENTPTR=.AZOFF(1);
        IF CURRENT>4999H THEN TEMPA=TENSCOMP(CURRENT) OR 8000H;
        ELSE TEMPA=CURRENT;
        CALL OUT$STRNG(TEMPA,INTERP,'Z');
    END;
    DO:                                     /*CASE REQUEST ELEV. OFFSET*/
        CURRENTPTR=.ELOFF(1);
        IF CURRENT>4999H THEN TEMPA=TENSCOMP(CURRENT) OR 8000H;
        ELSE TEMPA=CURRENT;
        CALL OUT$STRNG(TEMPA,INTERP,'V');
    END;
```

```
        DO;                                     /*CASE REQUEST DATE*/
            CURRENTPTR=.TEMP(1);
            TEMP(1)=NOT (IN(DAYLO)) AND 0F0H;
            TEMP(2)=IN(DAYHI);
            CALL OUT$STRNG(CURRENT,INTERP,'J');
        END;
    END;
END PRFRM$CMD;
CND$CMD:        PROCEDURE(INTERP);
    /*
    INTERPRET COMMANDS CONDITIONALLY ACCORDING TO CONTROLLER STATUS:
    AUTO OR IMMEDIATE MODE

    CND$CMD USES:  PRFRM$CMD,DECODE
    */
    DCL SET$MMODE  LIT 'CMDREG=5';
    DCL SET$INTRVL LIT 'CMDREG=3';
    DCL INTERP BYTE;
    IF DGTBUFPTR<>0 THEN
    VAL=DECODE(.DGTBUF,.DGTBUFPTR);  /*VAL HOLDS 4-DGT BCD SGN-MAG NO.*/
    ELSE VAL=0;
    DO CASE INTERP;
        DO;                 /*IMMEDIATE MODE*/
            IF STFLG THEN
                DO;
                    IF CMDREG>0FH THEN CALL PRFRM$CMD(CMDREG,INTERP);
                    ELSE
                        DO;
                            IF MMFLG THEN
                                DO;
                                    IF NOT SET$INTRVL THEN
                                    CALL PRFRM$CMD(CMDREG,INTERP);
                                END;
                            ELSE
                                DO CASE CMDREG;
                                /*IMMEDIATE MODE COMMANDS ALLOWED
                                DURING AN AUTORUN*/
                                    ;
                                    /*TIME OFFSET*/
                                    CALL PRFRM$CMD(CMDREG,INTERP);
                                    ;;
                                    /*ABORT RUN*/
                                    CALL PRFRM$CMD(CMDREG,INTERP);
                                    /*SET IMMEDIATE MODE*/
                                    CALL PRFRM$CMD(CMDREG,INTERP);
                                    ;;;;;;;
                                    /*SET AZIMUTH OFFSET*/
                                    CALL PRFRM$CMD(CMDREG,INTERP);
                                    /*SET ELEVATION OFFSET*/
                                    CALL PRFRM$CMD(CMDREG,INTERP);
                                END;
                        END;
                END;
            ELSE CALL PRFRM$CMD(CMDREG,INTERP);
        END;
```

```
        DO;                 /*AUTO MODE*/
            IF AUTOFLG THEN
                DO;
                    IF NOT (SET$MMODE) THEN CALL PRFRM$CMD(CMDREG,INTERP);
                END;
            ELSE
                DO;
                    /*PERFORM CMD. FOR A,E,I,O.  A,E ARE
                    UPDATED BUT NOT OUTPUT*/
                    IF CMDREG<4 THEN CALL PRFRM$CMD(CMDREG,INTERP);
                END;
        END;
        DO;                 /*SETUP RECORD*/
            IF NOT SET$MMODE THEN CALL PRFRM$CMD(CMDREG,INTERP);
        END;
    END;
    DGTBUFPTR=0;  /* SET UP FOR NEXT COMMAND*/
    CMDFLG=FALSE;
END CND$CMD;
PROC$REC:       PROCEDURE(INTERP);
    /*
    PROC$REC PROCESSES THE RECORD OF COMMANDS, DECODES COMMANDS
    AND NUMERIC DATA AND CAUSES EACH COMMAND IN THE RECORD TO BE
    SET UP FOR EXECUTION
    PROC$REC USES:  CND$CMD
    */
    DCL (INTERP,CMDCD,CHAR,INDEX) BYTE;
    DCL ALPHA LIT 'CHAR>=''A'' AND CHAR<=''Z'' OR (CHAR>140Q
    AND CHAR<173Q)';
    DO INDEX=0 TO 7;
        DGTBUF(INDEX)='0';  /*ASCII ZEROS TO DGTBUF*/
    END;
    IF INTERP=0 THEN
        DO;
            CHAR=MMIN(INSTPTR);
            INSTPTR=INSTPTR+1;
            IF ZERO THEN INFF=FALSE;
        END;
    ELSE
        DO;
            CHAR=DATABUF(DATASTPTR);
            DATASTPTR=DATASTPTR+1;
        END;
    DO WHILE CHAR<>':';
        CMDCD=-1;
        IF (CHAR='+') OR (CHAR='.') OR (CHAR='-') OR
        CHAR$IS$DIGIT THEN
            DO;
                IF DGTBUFPTR<8 THEN
                    DO;
                        DGTBUF(DGTBUFPTR)=CHAR;
                        DGTBUFPTR=DGTBUFPTR+1;
                    END;
            END;
```

```
        ELSE
            DO:
                IF ALPHA THEN
                    DO CASE CHAR AND 1FH:
                        :                   /*NON-ALPHA CHAR*/
                        CMDCD=0;            /*A-AZIMUTH*/
                        :
                        CMDCD=9;            /*C- TIME TO BUFFER*/
                        :
                        CMDCD=2;            /*E-ELEVATION*/
                        :
                        :
                        :
                        CMDCD=3;            /*I-INTERVAL*/
                        CMDCD=15;           /*J-JULIAN DATE*/
                        CMDCD=4;            /*K- ABORT RUN*/
                        RQSTFLG=TRUE;       /*REQUEST*/
                        CMDCD=5;            /*M-IMMED. MODE*/
                        :
                        CMDCD=1;            /*O-TIMEOFFSET*/
                        :
                        :
                        CMDCD=6;            /*R-REPORT*/
                        CMDCD=7;            /*S-STATUS*/
                        CMDCD=8;            /*T-REAL TIME TO PDP*/
                        :
                        CMDCD=14;           /*V-ELEVATION OFFSET*/
                        CMDCD=10;           /*W-WAITMODE*/
                        :
                        :
                        CMDCD=13;           /*Z-AZIMUTH OFFSET*/
                    END;
            END;
        IF CHAR='$' THEN CMDCD=12;          /*CASE END-RUN */
        IF CMDFLG THEN
            DO:
                IF (CHAR=',') OR (CHAR='L') OR CMDCD<>-1 THEN
                CALL CND$CMD(INTERP);
                IF ABRTFLG THEN DO; ABRTFLG=FALSE; RETURN; END;
            END;
        IF CMDCD<>-1 THEN
            DO:
                CMDFLG=TRUE;
                IF RQSTFLG THEN CMDREG=CMDCD OR 10H;
                ELSE CMDREG=CMDCD;
                RQSTFLG=FALSE;
            END;
        IF ABRTFLG THEN DO; ABRTFLG=FALSE; RETURN; END;
        IF INTERP=0 THEN
            DO:
                CHAR=MMIN(INSTPTR);
                INSTPTR=INSTPTR+1;
                IF ZERO THEN INFF=FALSE;
            END;
```

```
        ELSE
            DO;
                CHAR=DATABUF(DATASTPTR);
                DATASTPTR=DATASTPTR+1;
            END;
    END;
    IF CMDFLG THEN
        DO;
            CALL CND$CMD(INTERP);
            IF ABRTFLG THEN DO; ABRTFLG=FALSE; RETURN; END;
        END;
END PROC$REC;
PROC$MM$REC:        PROCEDURE;
    /*PROCESS A MANUAL MODE RECORD
    PROC$MM$REC USES:  PROC$REC*/
    CALL PROC$REC(0);
    INSEM=INSEM-1;
    MMOUT(OUTNDPTR)=':';  /*  SHOW A NEW RECORD HAS BEEN*/
    OUTNDPTR=OUTNDPTR+1;  /* ADDED TO THE OUTPUT BUFFER     */
    IF ZERO THEN OUTFF=TRUE;
    OUTSEM=OUTSEM+1;
END PROC$MM$REC;
PROC$DATA$REC:        PROCEDURE;
    /*PROCESS A DATA RECORD FROM DATABUF
    PROC$DATA$REC USES:  PROC$REC*/
    CALL PROC$REC(1);
    RPRTBUF(RPRTBUFPTR)=':';
    RPRTBUFPTR=RPRTBUFPTR+1;
    RPRTRECTR=RPRTRECTR+1;
    DATARECTR=DATARECTR-1;
    RPRTFLG=FALSE;  /*PENDING REPORTS CANCELLED ON SETUP AUTORUN*/
    NXTRECSEM=NXTRECSEM-1;
END PROC$DATA$REC;
SETUP$REC:        PROCEDURE;
    /*
    SETUP$REC READS AND PERFORM THE FIRST RECORD OF A
    REMOTE RUN.  THIS RECORD SHOULD PREPOSITION THE ANTENNA
    AND SET THE PROPER INTERVAL.
    SETUP$REC USES:  PROC$REC
    */
    ELINC(0)=(ELINC(1):=(AZINC(0):=(AZINC(1):=0)));
    ELINC(2)=(AZINC(2):=0);
    AZOFF(0)=(AZOFF(1):=(AZOFF(2):=0));
    ELOFF(0)=(ELOFF(1):=(ELOFF(2):=0));
    ELHOLD(0)=(ELHOLD(1):=(ELHOLD(2):=0));
    AZHOLD(0)=(AZHOLD(1):=(AZHOLD(2):=0));
    ELOLD=(AZOLD:=0);
    RPRTFLG=FALSE;        /*TURN OFF OLD REPORT*/
    CALL PROC$REC(2);
    RPRTBUF(RPRTBUFPTR)=':';
    RPRTBUFPTR=RPRTBUFPTR+1;
    RPRTRECTR=1;        /* ALWAYS PRESET RPRTRECTR ON SETUP */
    DATARECTR=DATARECTR-1;
    NOTSETUPFLG=FALSE;
END SETUP$REC;
```

```
/************** END DECODE AND PROCESSING ************************/
/*                                                                */
/**********************BEGIN INTERRUPT ROUTINES********************/
/*                                                                */
PDP$SBC:        PROCEDURE (CHAR);
    DCL STX LIT '02H';
    DCL EOT LIT '04H';
    DCL CHAR BYTE;
    IF STXFLG THEN
        DO;
            IF CHAR=EOT THEN
                DO;
                    STXFLG=FALSE;
                    RETURN;
                END;
            IF CHAR=':' THEN DATARECTR=DATARECTR+1;
            DATABUF(DATANDPTR)=CHAR;
            DATANDPTR=DATANDPTR + 1;
        END;
    ELSE
        DO;
            IF CHAR=STX THEN
                DO;
                    STXFLG=TRUE;
                    RETURN;
                END;
            IF CHAR=':' THEN INSEM=INSEM + 1;
            MMIN(INNDPTR)=CHAR;
            INNDPTR=INNDPTR + 1;
            IF ZERO THEN INFF=TRUE;
        END;
END PDP$SBC;
PDP$SBC$ISR:        PROCEDURE INTERRUPT 4;
    /*PDP$SBC$ISR USES:  PDP$SBC*/
    DCL CHAR BYTE;
    DCL EOIC LIT '20H';
    DCL ICCP LIT '0DAH';
    DCL CTLX LIT '18H';
    DCL EXIT LIT '08';  /* ESCAPE TO MONITOR */
    CHAR=IN(RS232);
    IF CHAR=CTLX THEN
        DO;
            OUT(ICCP)=EOIC;
            GO TO EXIT;
        END;
    IF INFF AND (INSTPTR-INNDPTR<2) THEN
        DO;
            CALL OUT$CHAR(CTLS);
            CTLSFLG=TRUE;
        END;
    CALL PDP$SBC(CHAR);
    OUT(ICCP)=EOIC;        /*RESET INTERRUPT CHIP*/
END PDP$SBC$ISR;
```

```
SBC$PDP:        PROCEDURE;
    DCL CHAR BYTE;
    CHAR=MMOUT(OUTSTPTR);
    IF ENDOUTFLG THEN
        DO;
            ENDOUTFLG=FALSE;
            MMOUTBUSY=FALSE;
        END;
    ELSE    /*RESET REQUEST ONLY AFTER LAST CHAR TRANSMITTED*/
        DO;    /*HAVE NOT SEEN EOR YET, SEND NEXT CHAR*/
            IF CHAR=':' THEN
                DO;
                    OUTSEM=OUTSEM-1;
                    ENDOUTFLG=TRUE;
                END;
            OUT(RS232)=CHAR;
            OUTSTPTR=OUTSTPTR+1;
            IF ZERO THEN OUTFF=FALSE;
        END;
END SBC$PDP;
SBC$PDP$ISR:        PROCEDURE INTERRUPT 5;
    /* OUTPUT CHARACTER ON XMIT. BUF. EMPTY INTERRUPT*/
    DCL EOIC LIT '20H';
    DCL ICCP LIT '0DAH';
    CALL SBC$PDP;
    OUT(ICCP)=EOIC;         /*RESET INTERRUPT CHIP*/
END SBC$PDP$ISR;
TIMEQUAL:       PROCEDURE BYTE;
    /*TIMEQUAL RETURNS TRUE WHEN SET TIME = REAL TIME.
    TIMEQUAL USES:      */
    IF TSTART(0) <> IN(HR) THEN RETURN FALSE;
    IF TSTART(1) <> IN(MIN) THEN RETURN FALSE;
    IF TSTART(2) <> (NOT IN(SEC)) THEN RETURN FALSE;
    RETURN TRUE;
END TIMEQUAL;
TRANSFER$TIME:        PROCEDURE;
    /*TRANSFER TIME TO PDP11.
    TRANSFER$TIME USES:  BCDBYTASC,PDP$SBC*/
    DCL CTLT LIT '14H';
    DCL TX$NOT$RDY LIT '(TEST AND 1)=0';
    DCL CHECK$STATUS LIT 'TEST=IN(RS232$CTL)';
    DCL RX$RDY LIT '(TEST AND 2)<>0';
    DCL TIM (2) BYTE;
    DCL (TEMP,TEST,I,PTR) BYTE;
    DO I=0 TO 6;
        CHECK$STATUS;
```

```
        DO WHILE TX$NOT$RDY;
            IF RX$RDY THEN
            /*MONITOR AND INPUT CHAR.S FROM PDP
            WHILE WAITING FOR USART OUTPUT RDY*/
                DO;
                    TEMP=IN(RS232);
            /*HANDLE INPUT CHAR IN APPROPRIATE MANNER*/
                    CALL PDP$SBC(TEMP);
                END;
            CHECK$STATUS;
        END;
        DO CASE I;
            /*ASSUME UNPACKED DATA OF 24 BITS*/
            OUT(RS232)='T';
            DO;
                PTR=0;
                TEMP=IN(HR);
                CALL BCDBYTASC(.TIM..PTR,TEMP);
                OUT(RS232)=TIM(0);
            END;
            OUT(RS232)=TIM(1);
            DO;
                PTR=0;
                TEMP=IN(MIN);
                CALL BCDBYTASC(.TIM..PTR,TEMP);
                OUT(RS232)=TIM(0);
            END;
            OUT(RS232)=TIM(1);
            DO;
                PTR=0;
                TEMP=NOT IN(SEC);
                CALL BCDBYTASC(.TIM..PTR,TEMP);
                OUT(RS232)=TIM(0);
            END;
            OUT(RS232)=TIM(1);  /*OUTPUT LAST CHAR*/
        END;
    END;
    /*TURNOFF OUTPUT STREAM IF SYSTEM OUTPUT WAS NOT ACTIVE*/
    /*MMOUTBUSY=TRUE PREVENTS OVERRUNS ON OUTPUT */
    IF NOT MMOUTBUSY THEN
        DO;
            ENDOUTFLG=TRUE;
            MMOUTBUSY=TRUE;
        END;
    TMRQFLG=FALSE;
END TRANSFER$TIME;
TICK:       PROCEDURE INTERRUPT 3;
    /*TIMECODE INTERRUPT KEEPS REAL TIME.  TRANSFER REAL TIME
    AS REQUESTED ON SECOND TICK
    TICK USES:  TIMEQUAL,ALARM,TRANSFER$TIME*/
    DCL SCNDBIT LIT 'IN(0E4H) AND 01';
    DCL EOIC LIT '20H';
    DCL ICCP LIT '0DAH';
```

```
      RELOAD$CTR;
      NEWSEC=SCNDBIT;
      IF OLDSEC<>NEWSEC THEN
          DO;
              IF CNTDWNFLG THEN
                  DO;
                      IF TIMEQUAL THEN
                          DO;
                              STFLG=TRUE;
                              CNTDWNFLG=FALSE;
                              INTRVLCTR=0;
                              TURNONFLG=TRUE;
                          END;
                  END;
              IF TMRQFLG THEN CALL TRANSFER$TIME;
          END;
      OLDSEC=NEWSEC;
      IF STFLG THEN
          DO;
              IF TMDLY<>0 THEN TMDLY=TMDLY-1;
              ELSE
                  DO;
                      IF INTRVLCTR=0 THEN
                          DO;         /*READ RECORD TICK*/
                              INTRVLCTR=INTRVL-1;
                              IF DATARECTR<>0 THEN NXTRECSEM=NXTRECSEM+1;
                          END;
                      ELSE
                          DO;         /*INTERPOLATION TICK*/
                              INTRVLCTR=INTRVLCTR-1;
                              TICKSEM=TICKSEM+1;
                          END;
                  END;
          END;
      ELSE
          DO;
              IF AUTOFLG AND NOT CNTDWNFLG THEN
                  DO;
                      NOTSETUPFLG=TRUE;
                      CNTDWNFLG=TRUE;
                  END;

          END;
      OUT(ICCP)=EOIC;          /*RESET INTERRUPT CHIP*/
END TICK;
/********** END INTERRUPT ROUTINES **********/
```

```
/*                                                                    */
/********************* TASK PRIORITY RESOLVER ************************/
/*                                                                    */
STARTUP:
DCL FOREVER LIT 'WHILE 0=0';
DCL TXRDY LIT '(IN(0EFH) AND 1) = 1';
    CALL SET$INT;
    CALL ZERO$FLGS;
    CALL SET$IO;
    DO FOREVER;
        PRIORITY=0;
        IF RPRTFLG AND NOT STFLG AND NOT MMOUTBUSY THEN PRIORITY=4;
        IF INSEM<>0 THEN
            DO;
                IF OUTFF THEN
                    DO;
                        IF OUTSTPTR-OUTNDPTR>32 THEN PRIORITY=5;
                        /*PROC$MM$REC ONLY IF NO DANGER OF MMOUT OVRFLW*/
                    END;
                ELSE PRIORITY=5;
            END;
        IF MMOUTBUSY AND TXRDY THEN PRIORITY=7;
        ELSE
            DO;
                IF OUTSEM<>0 AND NOT PDPBSY THEN PRIORITY=6;
            END;
        IF CTLSFLG AND NOT INFF THEN PRIORITY=8;
        IF TICKSEM>0 THEN PRIORITY=9;
        IF DONEFLG THEN PRIORITY=10;
        IF NXTRECSEM <> 0 THEN PRIORITY =11;
        IF NOTSETUPFLG THEN PRIORITY=12;
        IF TURNONFLG THEN PRIORITY=13;
        DO CASE PRIORITY;
            ;
            ;
            ;
            ;
            CALL REPORT;
            CALL PROC$MM$REC;
            CALL OUT$REC;
            CALL SBC$PDP;
            DO;
                CALL OUT$CHAR(CTLQ);
                CTLSFLG=FALSE;
            END;
            DO;
                CALL INC$AZEL;
                TICKSEM=TICKSEM-1;
            END;
```

```
            DO;
                IF NXTRECSEM=0 THEN
                    DO;
                        DATASTPTR=0;
                        DONEFLG=FALSE;
                        CALL OUT$CHAR(CTLZ);
                    END;
            END;
            CALL PROC$DATA$REC;
            CALL SETUP$REC;
            DO;
                CALL OUT$CHAR(CTLA);
                TURNONFLG=FALSE;
            END;
        END;
    END;
EOF
```

APPENDIX F

Direct Communication with the Antenna Controller

F.0 Direct Communication with Antenna Controller

The user is usually isolated from direct communication with the antenna controller. The operator interacts with the graphics display on the PDP11, and this interaction is translated into a series of commands to the antenna controller by the PDP11 THREADER program.

A PDP11 assembly language program called 'SBC001' exists to provide the user a means of issuing commands directly to the microcomputer memory. The keyboard console becomes the medium for issuing commands. To utilize SBC001, type

R SBC001

then enter UUUUUU... until the SBC80/20 monitor sign-on message appears. The console is now in direct communication with the SBC80/20 monitor [see Ref. 1]. The monitor program resides in ROM on the SBC80/20 microcomputer card and serves as a bootstrap aid for the THREADER program. As the antenna controller program resides on the PDP11 disk, the program must be downloaded to the antenna controller memory. To download, enter

T

and the SBC001 program responds:

FILENAME=

at which point the name of the object file for the antenna controller program should be entered. Transfer takes several minutes and is terminated by two bell tones and the monitor prompt, '?'. To start the THREADER program, which begins at hex-location (4003) enter:

G4003

Once the THREADER program has begun, any valid antenna controller command given in section 6.0.2 may be entered. To return to SBC80/20 monitor, type '↑X'. The ↑X option is useful for system debug.

A listing of SBC001 is given in Appendix G. The monitor T-command is a non-standard modification to the SBC monitor, Version 1.2, to facilitate file transfer from the PDP11 to the SBC80 microcomputer(See Appendix H).

THREADER MICROCOMPUTER ANTENNA CONTROLLER

APPENDIX G

Listing of SBC001[Ref. 11]

```
; BLISS11 V.77267        Wednesday 1-Feb-78 10:58.57     SBCNEW.811      Page 1

:PDP10
;        0001    MODULE SBCCOM(ZIP,PIC)=
;        0002    BEGIN
;        0003    REQUIRE RT11.REQ;
:00100   0004    % RT11V2C.REQ VERSION 2.3 % SWITCHES NOLIST;
;        0559    BYTE OWN BUFC=2000],DONEFLG,RCVFLG,ENQFLG;
;        0560    OWN CHAR;
;        0561    BIND
;        0562        JSW =#000044,   !JOB STATUS WORD
;        0563        RCSR=#775620,   !RCVR STATUS REGISTER
;        0564        XCSR=#775624,   !TRANSMITTER STATUS REGISTER
;        0565        RBUF=#775622,   !RCVR BUFFER
;        0566        XBUF=#775626;   !TRANSMITTER BUFFER.
;        0567    MACRO
;        0568        LF=#012$,
;        0569        CR=#015$,
;        0570        OFF=0$,
;        0571        ON=1$,
;        0572        ENQ=05$,
;        0573        DTR=RCSR<1,1>$,
;        0574        CARDET=.RCSR<12,1>$,
;        0575        RTS=RCSR<2,1>$,
;        0576        TXRDY=.XCSR<7,1>$,
;        0577        RXRDY=.RCSR<7,1>$,
;        0578        CTS=.RCSR<13,1>$;
.SBTTL   GLOBAL ROUTINE SBCCOM=
;        0579    GLOBAL ROUTINE SBCCOM=
;        0580    BEGIN
;        0581    EXTERNAL RCVRINT,FILNAM;
;        0582    LABEL WRTLOOP,TFRLOOP;
;        0583    OWN CHARTT,BLKNO,WDCNT,BYTCNT,LOOPMAX,AREA[#10],DBLK;
;        0584    LOCAL PTR,TEMP;
;        0585    BYTE OWN FLAG;
;        0586    MACRO FILLBUF(LOOPST,BLKCNT,BUFLOOP)=
;    M   0587    BEGIN
;    M   0588        LABEL BUFLOOP;
;    M   0589        PTR=LOOPST;
;    M   0590        LOOPMAX=LOOPST+#1000;
;    M   0591        WDCNT=256;
;    M   0592        BUFLOOP:    BEGIN
;    M   0593        WHILE .PTR LSS .LOOPMAX DO
;    M   0594            BEGIN
;    M   0595                IF .DONEFLG THEN
;    M   0596                    (WDCNT=(.PTR-LOOPST+1)^-1;
;    M   0597                    LEAVE BUFLOOP)
;    M   0598                ELSE
;    M   0599                    (IF .RCVFLG THEN
;    M   0600                        (RCVFLG=OFF;
;    M   0601                        BUFC.PTR]=.CHAR;
;    M   0602                        PTR=.PTR+1));
;    M   0603            END;
;    M   0604        END;
;    M   0605            ?.WAIT(1);
;    M   0606            ?.WRITE(AREA,1,BUF[LOOPST],,WDCNT,BLKCNT);
;    M   0607            IF .DONEFLG THEN LEAVE WRTLOOP;
;        0608    END$;
;        0609    MACRO TFRERR=
;    M   0610    BEGIN
;    M   0611        IF .#52<0,8> NEQ 0 THEN ?.PRINT(PLIT('?JTM/TFRERR/'));
;    M   0612        LEAVE TFRLOOP;
;        0613    END$;
;        0614    !
;        0615    !  LINK TO MACHINE LANGUAGE
;        0616    !
;        0617    OPLABEL $NOCHAR;
;        0618    OPCODE $BR=BR;
;        0619    !
;        0620    !  TURN ON DL11E
;        0621    !
;        0622        DTR=ON;
;        0623        RTS=ON;
;        0624    !
;        0625    !  NO ECHO TO DEC-WRITER
;        0626    !
;        0627        JSW<12,1>=1;
;        0628    !
;        0629    !  LOAD DL11E INTERRUPT VECTOR
;        0630    !
;        0631        #350=RCVRINT;
;        0632        #352=#200;  !SET INTERRUPT PRIORITY OF RCVRINT TO BR4.
;        0633        RCSR<6,1>=ON;       !ENABLE RCVR INTERRUPT
```

```
;       0634    !
;       0635    !       FETCH DK HANDLER
;       0636    !
;       0637        ?.FETCH(%40000,PLIT(RAD50'DK '));
;       0638        ENQFLG=OFF;
;       0639        WHILE 1 DO
;       0640            BEGIN
;       0641                BLKNO=0;
;       0642                IF .RCVFLG THEN
;       0643                    (RCVFLG=OFF;
;       0644                    ?.TTYOUT(.CHAR);
;       0645                    DONEFLG=OFF;
;       0646                    SELECT .CHAR<0,8> OF NSET
;       0647                        '*':        (IF (.FLAG EQL 'T') OR (.FLAG EQL 'W') THEN ?.CLOSE(1);
;       0648                                    FLAG=0);
;       0649                        CR :        BEGIN
;       0650                            SELECT .FLAG OF NSET
;       0651                                'T':  BEGIN
;       0652                                    TFRLOOP:        BEGIN
;       0653                                    WHILE 1 DO
;       0654                                        BEGIN
;       0655                                            ?.READW(AREA,1,BUF,256,,BLKNO,TFRERR);
;       0656                                            BYTCNT=.VREG*1;

; BLISS11 V.77267       Wednesday 1-Feb-78 10:55.21     SBCNEW.B11      Page 1-2

;       0657                                            PTR=0;
;       0658                                            BLKNO=.BLKNO+1;
;       0659                                            WHILE .PTR LSS .BYTCNT DO
;       0660                                                BEGIN
;       0661                                                    IF .ENQFLG EQL ON THEN
;       0662                                                        (WHILE NOT TXRDY DO;
;       0663                                                        TEMP=.BUF[.PTR];
;       0664                                                        PTR=.PTR+1;
;       0665                                                        XBUF=.TEMP;
;       0666                                                        ENQFLG=OFF);
;       0667                                                    IF .DONEFLG EQL ON THEN LEAVE TFRLOOP;
;       0668                                                END;
;       0669                                        END;
;       0670                                    END;
;       0671                                    FLAG=0;
;       0672                                    ?.CLOSE(1);
;       0673                                    ?.PRINT(PLIT('?G?G'));
;       0674                                END;
;       0675                            'W':        BEGIN
;       0676                                WRTLOOP:        BEGIN
;       0677                                WHILE 1 DO
;       0678                                    BEGIN
;       0679                                        FILLBUF(0,,BLKNO,LOOP1);
;       0680                                        BLKNO=.BLKNO+1;
;       0681                                        FILLBUF(%1000,,BLKNO,LOOP2);
;       0682                                        BLKNO=.BLKNO+1;
;       0683                                    END;
;       0684                                END;
;       0685                                FLAG=0;
;       0686                                ?.CLOSE(1);
;       0687                                ?.PRINT(PLIT('?G?G'));
;       0688                                DONEFLG=OFF;
;       0689                            END;
;       0690                        TESN;
;       0691                    END;
;       0692                TESN);
;       0693            CHARTT<0,8>=?.TTINR(%BR(%NOCHAR));
;       0694            SELECT .CHARTT<0,8> OF NSET
;       0695                LF :    %BR(%NOCHAR);   !DON'T TRANSMIT LF'S
;       0696                'T':    FLAG='T';
;       0697                'W':    FLAG='W';
;       0698                CR :    SELECT .FLAG OF NSET
;       0699                            'T':
;       0700                                ?.LOOKUP(AREA,1,FILNAM(PLIT(RAD50'DK FTN4  DAT')));
;       0701
;       0702                            'W':
;       0703                                (DBLK=FILNAM(0);
;       0704                                IF .DBLK NEQ 0 THEN ?.ENTER(AREA,1,,DBLK,*100)
;       0705                                ELSE FLAG=0);
;       0706                        TESN;
;       0707            TESN;
;       0708            WHILE NOT TXRDY DO;
;       0709            XBUF=.CHARTT;
;       0710            %NOCHAR:    ;
;       0711        END;
;       0712    END;
```

```
; BLISS11 V.77267        Wednesday 1-Feb-78 10:57.53     SBCNEW.B11      Page 1-3

;       0709                XBUF=.CHARTT;
;       0710                $NOCHAR:    ;
;       0711            END;
;       0712    END;
.TITLE SBCCOM
.CSECT SBCC.C

R$0=%0
R$1=%1
R$2=%2
R$3=%3
R$4=%4
R$5=%5
SP=%6
PC=%7
CLVC=243


SBCCOM:
        BIS     #10000,@#44
        MOV     #RCVRINT,@#350
        MOV     #200,@#352
        BISB    #186,@#175620
        MOV     #P$AAB,R$0
        MOV     #40000,-(SP)
        EMT     343
        CLRB    ENDFLG
L$12:   CLR     BLKNO
        BITB    #1,@#RCVFLG
        BNE     L$68
L$69:   JMP     L$34
L$68:   CLRB    RCVFLG
        MOV     @#CHAR,R$0
        BIC     #177400,R$0
U$167:  EMT     341
        BLO     U$167
        CLRB    DONEFLG
        CLR     R$2
        BISB    @#CHAR,R$2
        CMP     R$2,#43
        BNE     L$15
        CMPB    @#FLAG,#124
        BEQ     L$16
        CMPB    @#FLAG,#127
        BNE     L$18
L$16:   MOV     #3001,R$0
        EMT     374
L$18:   CLRB    FLAG
L$15:   CMP     R$2,#15
        BNE     L$69
        CLR     R$2
        BISB    @#FLAG,R$2
; BLISS11 V.77267        Wednesday 1-Feb-78 11:00.48     SBCNEW.B11      Page 1-4

        CMP     R$2,#124
        BNE     L$20
L$21:   MOV     #4001,AREA
        MOV     @#BLKNO,AREA+2
        MOV     #BUF,AREA+4
        MOV     #400,AREA+6
        CLR     AREA+10
        MOV     #AREA,R$0
        EMT     375
        BMIS    U$170
        TSTB    @#52
        BEQ     TFRLOOP
        MOV     #P$AAC,R$0
        EMT     351
        BR      TFRLOOP
E$3:
U$170:  MOV     R$0,BYTCNT
        ASL     BYTCNT
        CLR     R$1
        INC     BLKNO
L$24:   CMP     R$1,@#BYTCNT
        BGE     L$21
        CMPB    @#ENDFLG,#1
        BNE     L$28
```

```
L$27:   TSTB    @#175624
        BPL     L$27
        CLR     R$3
        BISB    BUF(R$1),R$3
        INC     R$1
        MOV     R$3,@#175626
        CLRB    ENQFLG
L$28:   CMPB    @#DONEFLG,#1
        BNE     L$24
TFRLOOP:
        CLRB    FLAG
        MOV     #3001,R$0
        EMT     374
        MOV     #P$AAD,R$0
        EMT     351
L$20:   CMP     R$2,#127
        BNE     L$69
L$35:   CLR     R$1
        MOV     #1000,LOOPMAX
        MOV     #400,WDCNT
L$36:   CMP     R$1,@#LOOPMAX
        BGE     LOOP1
        BITB    #1,@#DONEFLG
        BEQ     L$39
        MOV     R$1,R$0
        INC     R$0
        ASR     R$0
        MOV     R$0,WDCNT
; BLIS11 V.77267        Wednesday 1-Feb-78 11:00.44     SBCNEW.811      Page 1-5

        BR      LOOP1
L$39:   BITB    #1,@#RCVFLG
        BEQ     L$36
        CLRB    RCVFLG
        MOVB    @#CHAR,BUF(R$1)
        INC     R$1
        BR      L$36
LOOP1:  MOV     #1,R$0
        EMT     374
        MOV     #4401,AREA
        MOV     @#BLKNO,AREA+2
        MOV     #BUF,AREA+4
        MOV     @#WDCNT,AREA+6
        MOV     #1,AREA+10
        MOV     #AREA,R$0
        EMT     375
        BITB    #1,@#DONEFLG
        BNE     WRTLOOP
        INC     BLKNO
        MOV     #1000,R$1
        MOV     #2000,LOOPMAX
        MOV     #400,WDCNT
L$46:   CMP     R$1,@#LOOPMAX
        BGE     LOOP2
        BITB    #1,@#DONEFLG
        BEQ     L$49
        MOV     R$1,R$0
        SUB     #777,R$0
        ASR     R$0
        MOV     R$0,WDCNT
        BR      LOOP2
L$49:   BITB    #1,@#RCVFLG
        BEQ     L$46
        CLRB    RCVFLG
        MOVB    @#CHAR,BUF(R$1)
        INC     R$1
        BR      L$46
LOOP2:  MOV     #1,R$0
        EMT     374
        MOV     #4401,AREA
        MOV     @#BLKNO,AREA+2
        MOV     #BUF+1000,AREA+4
        MOV     @#WDCNT,AREA+6
        MOV     #1,AREA+10
        MOV     #AREA,R$0
        EMT     375
        BITB    #1,@#DONEFLG
        BNE     WRTLOOP
        INC     BLKNO
        BR      L$35
WRTLOOP:
        CLRB    FLAG
```

```
: BLISS11 V.77267        Wednesday 1-Feb-78 11:00.47     SBCNEW.B11      Page 1-6

        MOV     #3001,R$0
        EMT     374
        MOV     #P$AAE,R$0
        EMT     351
        CLRB    DONEFLG
L$34:   EMT     340
        BLO     $NOCHAR
E$11:
U$186:  MOVB    R$0,@#CHARTT
        CLR     R$0
        BISB    @#CHARTT,R$0
        CMP     R$0,#12
        BEQ     $NOCHAR
        CMP     R$0,#124
        BNE     L$57
        MOVB    #124,FLAG
L$57:   CMP     R$0,#127
        BNE     L$58
        MOVB    #127,FLAG
L$58:   CMP     R$0,#15
        BNE     L$61
        CLR     R$2
        BISB    @#FLAG,R$2
        CMP     R$2,#124
        BNE     L$60
        MOV     #401,AREA
        MOV     #P$AAF,-(SP)
        JSR     PC,@#FILNAM
        MOV     R$0,AREA+2
        CLR     AREA+4
        MOV     #AREA,R$0
        EMT     375
        TST     (SP)+
L$60:   CMP     R$2,#127
        BNE     L$61
        CLR     -(SP)
        JSR     PC,@#FILNAM
        MOV     R$0,DBLK
        BEQ     L$63
        MOV     #1001,AREA
        MOV     R$0,AREA+2
        MOV     #100,AREA+4
        CLR     AREA+6
        MOV     #AREA,R$0
        EMT     375
        BR      L$64
L$63:   CLRB    FLAG
L$64:   TST     (SP)+
L$61:   TSTB    @#175624
        BPL     L$61
        MOV     @#CHARTT,@#175626
$NOCHAR:
: BLISS11 V.77267        Wednesday 1-Feb-78 11:00.49     SBCNEW.B11      Page 1-7

        JMP     L$12

: ROUTINE SIZE:  339
```

```
; BLISS11 V.77267        Wednesday 1-Feb-78 11:00.49     SBCNEW.811      Page 1-7

        JMP     L$12

; ROUTINE SIZE:  339


.CSECT SBCC.0
LOOPMAX: .=.+2
CHARTT: .=.+2
BYTCNT: .=.+2
BLKNO: .=.+2
DBLK: .=.+2
FLAG: .=.+1
.EVEN
AREA: .=.+20
WDCNT: .=.+2
.CSECT SBCC.P
        +1
P$AAD:  .WORD  3407
        +1
P$AAE:  .WORD  3407
        +4
P$AAF:  .WORD  15270,24256,152200
        .WORD  14474
        +5
P$AAC:  .WORD  6412,52057,51106
        .WORD  51105,27522
        +1
P$AAB:  .WORD  15270
.GLOBL RCVRINT
.GLOBL FILNAM


.SBTTL  GLOBAL ROUTINE FILNAM(DFLTADR)=
;       0713    GLOBAL ROUTINE FILNAM(DFLTADR)=
;       0714    BEGIN
;       0715    EXTERNAL !RAD50;
;       0716    OPCODE $JSR=JSR;
;       0717    BYTE OWN ASCIIBUF[=20];
;       0718    OWN CHARF,RAD50BUF[=10];
;       0719    LOCAL I,DSKAD;
;       0720        ASCIIBUF[0]='D';
;       0721        ASCIIBUF[1]='K';
;       0722        INCR I FROM 2 TO =17 BY 1 DO ASCIIBUF[.I]=' ';
;       0723        ?.PRINT(PLIT('?M?JFILENAME',')='+=200+8));
;       0724        I=2;
;       0725        DO
;       0726            (CHARF=?.TTYIN;
;       0727            SELECT .CHARF<0,8> OF NSET
;       0728                '.':        (?.TTYOUT(.CHARF);I=8);
;       0729                '^J':       ;
;       0730                '^M':       ;
; BLISS11 V.77267        Wednesday 1-Feb-78 11:01.16     SBCNEW.811      Page 1-8

;       0731                OTHERWISE:  (?.TTYOUT(.CHARF);ASCIIBUF[I=.I+1]=.CHARF<0,8>);
;       0732            TESN)
;       0733        UNTIL .CHARF<0,8> EQL '^M';
;       0734        IF .I LEQ 2 THEN .DFLTADR !DEFAULT IF NO FILENAME
;       0735        ELSE
;       0736            (R5=PLIT(3,PLIT(12),ASCIIBUF,RAD50BUF); !FORTRAN LINK TO !RAD50
;       0737            $JSR(PC,!RAD50);
;       0738            RAD50BUF)
;       0739    END;
```

```
.CSECT SBCC.C

FILNAM:
        MOV     R$1,-(SP)
        MOV     R$2,-(SP)
        MOV     R$5,-(SP)
        MOVB    #104,ASCIIBUF
        MOVB    #113,ASCIIBUF+1
        MOV     #2,R$5
L$76:   MOVB    #40,ASCIIBUF(R$5)
        INC     R$5
        CMP     R$5,#17
        BLE     L$76
        MOV     #P$AAG,R$0
        EMT     351
        MOV     #2,R$5
L$78:
U$204:  EMT     340
        BLO     U$204
        MOV     R$0,CHARF
        CLR     R$2
        BISB    R$0,R$2
        CLR     R$1
        CMP     R$2,#56
        BNE     L$79
        BIC     #177400,R$0
U$205:  EMT     341
        BLO     U$205
        MOV     #10,R$5
        INC     R$1
L$79:   CMP     R$2,#12
        BNE     L$80
        INC     R$1
L$80:   CMP     R$2,#15
        BNE     L$81
        INC     R$1
L$81:   TST     R$1
        BGT     L$82
        MOV     @#CHARF,R$0
        BIC     #177400,R$0
U$207:  EMT     341
        BLO     U$207
```



```
        INC     R$5
        MOVB    @#CHARF,ASCIIBUF(R$5)
L$82:   CMPB    @#CHARF,#15
        BNE     L$78
        CMP     R$5,#2
        BGT     L$85
        MOV     10(SP),R$0
        BR      L$86
L$85:   MOV     #P$AAH,R$5
        JSR     PC,@#IRAD50
        MOV     #RAD50BUF,R$0
L$86:   MOV     (SP)+,R$5
        MOV     (SP)+,R$2
        MOV     (SP)+,R$1
        RTS     PC

; ROUTINE SIZE:  80


.CSECT SBCC.D
ASCIIBUF: .=.+20
CHARF:  .=.+2
RAD50BUF: .=.+20
.CSECT SBCC.P
        +6
P$AAG:  .WORD  5015,44506,42514
        .WORD  40516,42515,100075
        +4
P$AAH:  .WORD  3,P$AAI,ASCIIBUF
        .WORD  RAD50BUF
        +1
P$AAI:  .WORD  14
.GLOBL IRAD50
```

```
.SBTTL  GLOBAL ROUTINE INTERRUPT RCVRINT=
;       0740    GLOBAL ROUTINE INTERRUPT RCVRINT=
;       0741    BEGIN
;       0742        IF RXRDY THEN
;       0743            (CHAR=.RBUF<0,8>;
;       0744            IF .CHAR<0,8> EQL ENQ THEN ENQFLG=ON
;       0745            ELSE
;       0746                (IF .CHAR<0,8> EQL '??' THEN DONEFLG=ON;
;       0747                RCVFLG=ON));
;       0748    END;

.CSECT SBCC.C

RCVRINT:
        TSTB    @#175620
        BPL     L$92

; BLIS11 V.77267        Wednesday 1-Feb-78 11:02.24     SBCNEW.B11      Page 1-10

        CLR     CHAR
        MOVB    @#175622,CHAR
        CMPB    @#CHAR,#5
        BNE     L$91
        MOVB    #1,ENQFLG
        RTI
L$91:   CMPB    @#CHAR,#77
        BNE     L$94
        MOVB    #1,DONEFLG
L$94:   MOVB    #1,RCVFLG
L$92:   RTI

; ROUTINE SIZE:  27




;       0749    END
;       0750    ELUDOM


.CSECT SBCC.0
ENQFLG: .=.+1
.EVEN
CHAR: .=.+2
DONEFLG: .=.+1
RCVFLG: .=.+1
BUF: .=.+2000
.GLOBL RCVRINT
.GLOBL SBCCOM
.GLOBL FILNAM









; Size:  446+578
; Run Time:  53 Seconds
; Core Used:  25K
; Compilation Complete

.END
```

THREADER MICROCOMPUTER ANTENNA CONTROLLER

APPENDIX H

Modifications to SBC-Monitor, Version 1.2

THREADER MICROCOMPUTER ANTENNA CONTROLLER
SBC Monitor Modifications

H.0 Modifications to SBC-Monitor

H.1 Addition of T Command

The SBC monitor was modified to include the T command. The format of the T command is:

Thhhh

where 'hhhh' is an optional hex-offset. The T command enables the downloading of a hex object files from the PDP11 to the microcomputer memory. This is accomplished by a software request-response on a character by character basis. When the microcomputer is ready, an 'ENQ' is transmitted and the microcomputer enters a wait loop. The PDP11 program, SBC001, detects the ENQ and responds with the next character from the hex object file. The hex object file contains load address information within each record. The hex offset is added to the load address to determine the actual load address; if the offset is not prescribed, 0 is assumed.

The following shows the modifications to Version 1.2 of the SBC monitor to include the T command. Code lines in Version 1.2 are prefixed by '1)'; those in the modified monitor(denoted Version 1.2X) are prefixed by '2)'.

```
1)1             TITLE '80/20 MONITOR, V 1.2, 12 JLY 77'
2)1             TITLE '80/20 MONITOR, V 1.2X, 20 DEC 77'

1)1     ;                               VERSION 1.2
1)      ;                               12 JLY 1977
1)      ;
*****
2)1     ;                               VERSION 1.2X
2)      ;                               20 DEC 77
2)      ;
**********
1)1             MVI     C,'.'   ; PROMPT CHARACTER TO C
1)              CALL    ECHO    ; SEND PROMPT CHARACTER TO USER TERMINAL
*****
2)1             XRA     A
2)              STA     TCFLG   ; ZERO TCFLG--DEFAULT TO TTY MODE
2)              MVI     C,'?'   ; PROMPT CHARACTER TO C
2)              CALL    ECHO    ; SEND PROMPT CHARACTER TO USER TERMINAL
**********
1)1     ; CALLS: GETCH,ECHO,CO,RICH,BYTE
1)      ; DESTROYS: A,B,C,D,E,H,L,F/F'S
*****
2)1     ; CALLS: GETCH,ECHO,RICH,BYTE
2)      ; DESTROYS: A,B,C,D,E,H,L,F/F'S
**********
```

```
1)1     ; FUNCTION WCMD
*****
2)1     ; FUNCTON TCMD
2)      ; INPUTS:  NONE
2)      ; OUTPUTS:  NONE
2)      ; CALLS:  GETHX,ECHO,ERROR,RICH,BYTE
2)      ; DESTROYS:  A,B,C,D,E,H,L,F/F'S
2)      ; DESCRIPTION:  TCMD IMPLEMENTS READ OF A HEX-FILE FROM THE PDP11
2)      ;               UNDER SOFTWARE CONTROL.  THE T-COMMAND IS THHHH,
2)      ;               WHERE 'HHHH' IS AN OPTIONAL HEX OFFSET, ASSUMED 0
2)      ;               IF OMITTED.  THE OFFSET IS ADDED TO THE RECORD
2)      ;               ADDRESSES IN THE HES-FILE.
2)      ;
2)      TCMD:
2)              MVI     A,80H
2)              STA     TCFLG     ;SET TCFLG
2)              CALL    GETHX     ;GET A NUMBER TO BC
2)              JC      TCM04
2)              XRA     A         ;IF NO NUMBER IN INPUT STREAM
2)              MOV     B,A       ;ZERO BC
2)              MOV     C,A
2)      TCM04:
2)              PUSH    B         ;SAVE OFFSET
2)              MOV     C,D       ;GET DELIMITER
2)              CALL    ECHO      ;ECHO IT
2)              MOV     A,C
2)              CPI     CR        ;IF DELIMITER NOT CR
2)              JNZ     ERROR     ;RETURN ERROR, 'GETCM' CLEARS STACK
2)      TCM05:


2)              CALL    RICH      ;ELSE CONTINUE.
2)              CPI     ':'       ;LOOK FOR RECORD START
2)              JNZ     TCM05
2)              XRA     A         ;CLEAR A
2)              MOV     D,A       ;INITIALIZE D FOR CHECKSUM
2)              CALL    BYTE      ;READ RECORD LENGTH FROM INPUT
2)              JZ      GETCM     ;IF ZERO-DONE, 'GETCM' CLRS. STK.
2)              MOV     E,A       ;SAVE RECORD LENGTH IN E
2)              CALL    BYTE      ;MSB OF LOAD ADDRESS
2)              MOV     H,A       ;TO H
2)              CALL    BYTE      ;LSB OF LOAD ADDRESS
2)              MOV     L,A       ;TO L
2)              POP     B         ;GET OFFSET
2)              DAD     B         ;ADD OFFSET TO RECORD ADDRESS
2)              PUSH    B         ;SAVE OFFSET
2)              CALL    BYTE      ;GET RECORD TYPE
2)              MOV     C,E       ;RECORD LENGTH TO C
```

```
2)      TCM10:
2)              CALL    BYTE    ;GET DATA BYTE FROM TAPE
2)              MOV     M,A     ;STORE AND
2)              INX     H       ;INCREMENT POINTER
2)              DCR     E       ;DECREMENT RECORD LENGTH
2)              JNZ     TCM10   ;LOOP UNTIL DONE
2)              CALL    BYTE    ;READ CHECKSUM
2)              JNZ     ERROR   ;CHECKSUM ERROR IF NOT ZERO
2)              JMP     TCM05   ;GET ANOTHER RECORD
2)      ;
2)      ;
2)      ;**************************************************************************
2)      ;
2)      ;
2)      ; FUNCTION WCMD
**********
1)1     ; FUNCTION: GETHX
****
2)1             ORG     400H    ;REMAINDER OF MONITOR WILL GO ON UNIQUE 2708
2)      ;
2)      ;
2)      ;**************************************************************************
2)      ;
2)      ;
2)      ; FUNCTION: GETHX
**********
1)1             XRA     A       ; ZERO A
1)              STC             ; SET CARRY INDICATING TIMEOUT ERROR
****
2)1     RI14:   XRA     A       ; ZERO A
2)              STC             ; SET CARRY INDICATING TIMEOUT ERROR
**********
1)1             CALL    RI      ; READ A CHARACTER FROM TAPE
1)              JC      ERROR   ; JUMP IF READER TIMEOUT ERROR
1)              ANI     PRTY0   ; REMOVE PARITY BIT
****
2)1             LDA     TCFLG
2)              RLC             ; TEST FOR A TI OR RI CHARACTER READ
2)              CNC     RI      ; TCFLG CLEAR-RI
2)              LDA     TCFLG
2)              RLC
2)              CC      TI      ; TCFLG SET-INPUT FROM PDP11-TI
2)      RIRET:  JC      ERROR   ; JUMP IF READER TIMEOUT ERROR
2)              ANI     PRTY0   ; REMOVE PARITY BIT
**********
1)      ; FUNCTION: VALDG
****
2)1     ; FUNCTION:  TI
2)      ; INPUTS:  NONE
2)      ; OUTPUTS:  NONE
2)      ; CALLS:  DELAY
2)      ; DESTROYS:  A,F/F'S
```

```
2)      ; DESCRIPTION:  ENABLES SINGLE CHARACTER TRANSFER VIA SOFTWARE
2)      ;               REQUEST.  THE SBC80/20 OUTPUTS 'ENQ' AND WAITS
2)      ;               250 MSEC. FOR A CHARACTER RETURN.
2)      ;
2)      ENQ     EQU     05      ;ENQUIRY REQUESTS CHARACTER FROM PDP11
2)      TI:
2)              PUSH     B
2)      TI05:   IN       CNCTL      ; MAKE SURE NO CHARACTERS ARE IN
2)              ANI      TRDY       ; TRANSMITTER BUFFER.
2)              JZ       TI05       ; TI05 LOOP WAITS FOR TX BUFFER READY
2)              MVI      A,ENQ      ; TX BUFFER EMPTY-->OK TO TRANSMIT
2)              OUT      CNOUT      ; SEND ACKNOWLEDGE
2)      TI09:   MVI      B,250      ; SET WAIT LOOP FOR 250 MSEC.
2)              JMP     RI10    ;WAIT FOR CHARACTER RETURN
2)      ;
2)      ;
2)      ;**************************************************************************
2)      ;
2)      ;
2)      ; FUNCTION: VALDG
**********
1)1             DB      CR,LF,'80/20 MONITOR   V 1.2',CR,LF
1)      LSGNON  EQU     $-SGNON ; LENGTH OF SIGNON MESSAGE
****
2)1             DB      CR,LF,'80/20 MONITOR   V 1.2X',CR,LF
2)      LSGNON  EQU     $-SGNON ; LENGTH OF SIGNON MESSAGE
**********
1)1             DW   NCMD
1)              DW      XCMD
****
2)1             DW      TCMD    ;ADD TCMD TO COMMAND ADDRESS TABLE
2)              DW      NCMD
2)              DW      XCMD
**********
1)1             DB   'N'
1)      NCMDS   EQU     $-CTAB  ; NUMBER OF VALID COMMANDS
****
2)1             DB      'N'
2)              DB      'T'     ; ADD 'T' TO COMMAND CHARACTER TABLE


2)      NCMDS   EQU     $-CTAB  ; NUMBER OF VALID COMMANDS
**********
1)1     TEMP:   DB   0   ; TEMPORARY MONITOR CELL
1)      ;
****
2)1     TEMP:   DB      0       ; TEMPORARY MONITOR CELL
2)      TCFLG:  DB      0       ; FLAG TO CHOOSE TTY OR PDP11 READ OF HEX FILE
2)                              ; IF TCFLG SET READ PDP11, READ TTY(DEFAULT MODE)
2)      OFSTFLG:        DB      0       ;FLAG TO INDICATE OFFSET MUST BE COMPUTED
2)      ;
**********
```

H.2 INTSET

INTSET is an 8080 assembly language program which is called from the SET$INT procedure in the PL/M code. INTSET alters the interrupt jump table from state preset by the monitor. The service routines for interrupts 3,4,5 are set to TICK, PDPSBCISR and SBCPDPISR, respectively, replacing the monitor services routines. INTSET resides ar (3800) hex. The listing follows:

```
INTSET-->PGM. TO FILL INT. VCTS., SEQ.#1021, 13-MA          PAGE    1

       TITLE   'INTSET-->PGM. TO FILL INT. VCTS., SEQ.#1021, 13-MAR-79'
;**************************************************************************
;
;       INTSET IS CALLED FROM THE PL/M ROUTINE SETINT AND FILL THE
;              INTERRUPT VECTORS FOR THE 8259 SO THAT THE PL/M INTERRUPT
;              ROUTINES WILL BE ACCESSED.  THE ROUTINES OF INTEREST
;              ARE TICK,PDPSBCISR,SBCPDPISR.
;
;**************************************************************************
       LIST    X,M
       ORG     3800H   ;FIRST AVAILABLE RAM ON SBC80/20 BOARD.
;
;************************ INTERRUPT TABLE ADDRESSES ***********************
;
VECT5  EQU     3FF4H
VECT4  EQU     3FF0H
VECT3  EQU     3FECH
;
;************************** INTERRUPT PARAMETERS **************************
;
OCW1   EQU     0D9H
OST7OFF        EQU     0E0H
;
;****************************** MACROS ************************************
;
;       MVHX    MOVES BYTE ADDRESSED BY H TO LOCATION SPECIFIED BY
;               X(=D,B ONLY!).
;
;**************************************************************************
MVHX   MACRO   X
       MOV     A,M
       STAX    X
       INX     H
       INX     X
       ENDM
;**************************************************************************
;
```

```
THREADER MICROCOMPUTER ANTENNA CONTROLLER
SBC Monitor Modifications


;       LOAD FILLS AN INTERRUPT VECTOR JUMP INSTRUCTION INTO THE
;               INTERRUPT JUMP TABLE SPECIFIED BY ADR.
;
;**************************************************************************************
LOAD    MACRO   ADR,VECTOR
        LXI     D,ADR
        LXI     H,VECTOR
        MVHX    D
        MVHX    D
        MVHX    D
        ENDM
;
;**************************************************************************************
;


INTSET:
INTSET-->PGM. TO FILL INT. VCTS., SEQ.#1021, 13-MA          PAGE    2

        LOAD    VECT3,TICK
+       LXI     D,VECT3
+       LXI     H,TICK
+       MVHX    D
+       MOV     A,M
+       STAX    D
+       INX     H
+       INX     D
+       MVHX    D
+       MOV     A,M
+       STAX    D
+       INX     H
+       INX     D
+       MVHX    D
+       MOV     A,M
+       STAX    D
+       INX     H
+       INX     D
        LOAD    VECT4,PDPSBC
+       LXI     D,VECT4
+       LXI     H,PDPSBC
+       MVHX    D
+       MOV     A,M
+       STAX    D
+       INX     H
+       INX     D
+       MVHX    D
+       MOV     A,M
+       STAX    D
+       INX     H
+       INX     D
+       MVHX    D
```

THREADER MICROCOMPUTER ANTENNA CONTROLLER
SBC Monitor Modifications

```
+       MOV     A,M
+       STAX    D
+       INX     H
+       INX     D
        LOAD    VECT5,SBCPDP
+       LXI     D,VECT5
+       LXI     H,SBCPDP
+       MVHX    D
+       MOV     A,M
+       STAX    D
+       INX     H
+       INX     D
+       MVHX    D
+       MOV     A,M
+       STAX    D
+       INX     H
+       INX     D
+       MVHX    D
+       MOV     A,M

+       STAX    D
 INTSET-->PGM. TO FILL INT. VCTS., SEQ.#1021, 13-MA          PAGE    3

+       INX     H
+       INX     D
        RET
;
;*****************************************************************************
;
SBCPDP  JMP     59A4H
PDPSBC  JMP     5922H
TICK    JMP     5AE2H
;
;*****************************************************************************
;
        END
```

INDEX

AD-A075 106 NAVAL RESEARCH LAB WASHINGTON DC F/G 9/2
THREADER MICROCOMPUTER ANTENNA CONTROLLER.(U)
SEP 79 L E RUSSO
UNCLASSIFIED NRL-MR-4067 NL

2 OF 2
AD A075106

END
DATE FILMED
3-80
DDC